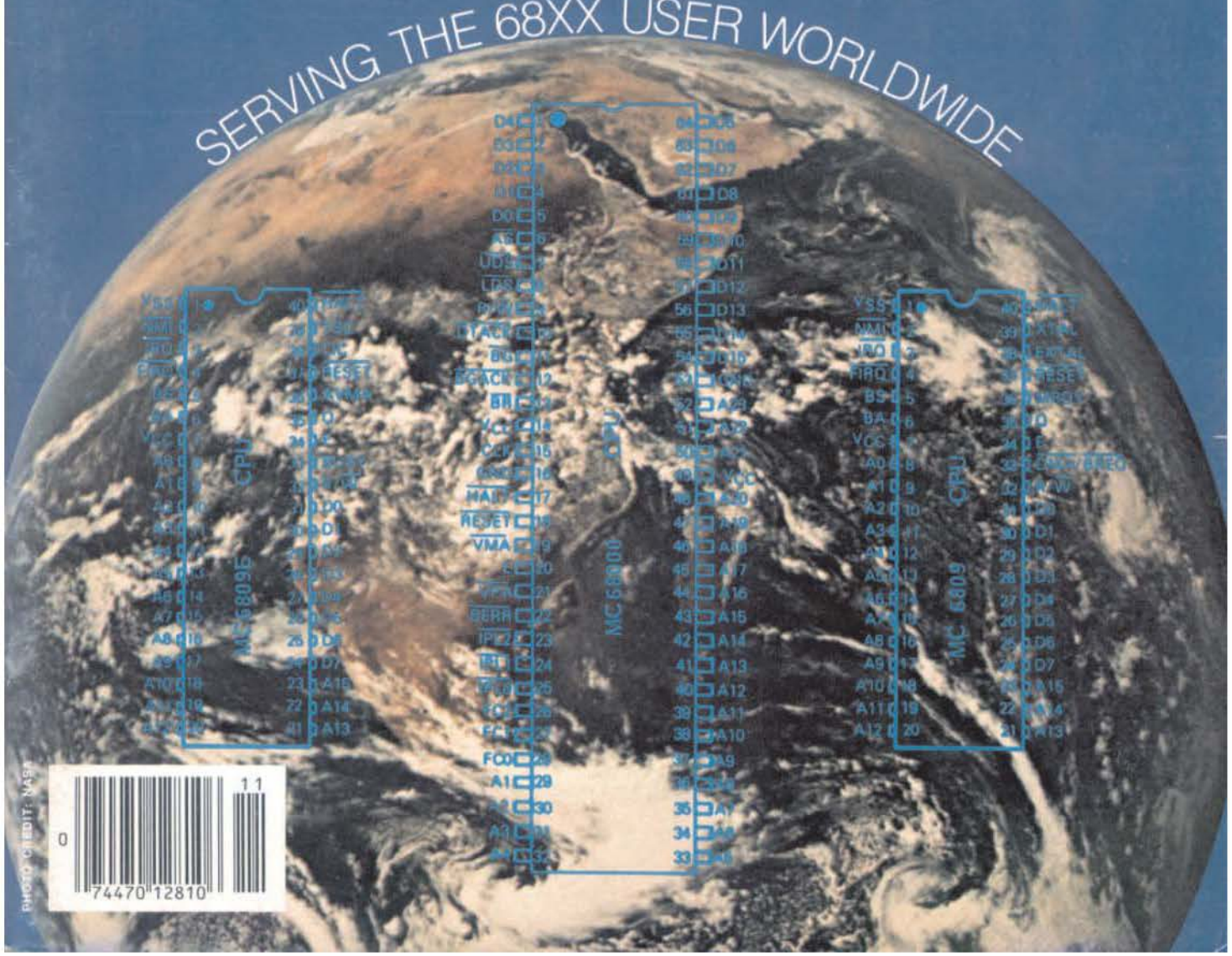
'68' MICRO JOURNAL

$2.95 USA

Australia	A $ 4.75	New Zealand	NZ $ 6.90
Singapore	S $ 9.45	Hong Kong	H $23.50
Malaysia	M $ 9.45	Sweden	30:-SEK

VOLUME VI ISSUE XI • Devoted to the 68XX User • November 1984

"Small Computers Doing Big Things"



0 74470 12810 11

WE DON'T PLAY GAMES

X-12+ A SERIOUS COMPUTER IN A DESKTOP PACKAGE

Multiprocessor Technology - Combination of 8, 16 and 32 bit types

1.0 Megabyte Memory - Insures no limitation on programs

"Winchester" Disk System - Fast response, large storage capacity

UniFlex* Operating System - The standard of comparison

Hardware Floating Point - Unmatched speed in a small system

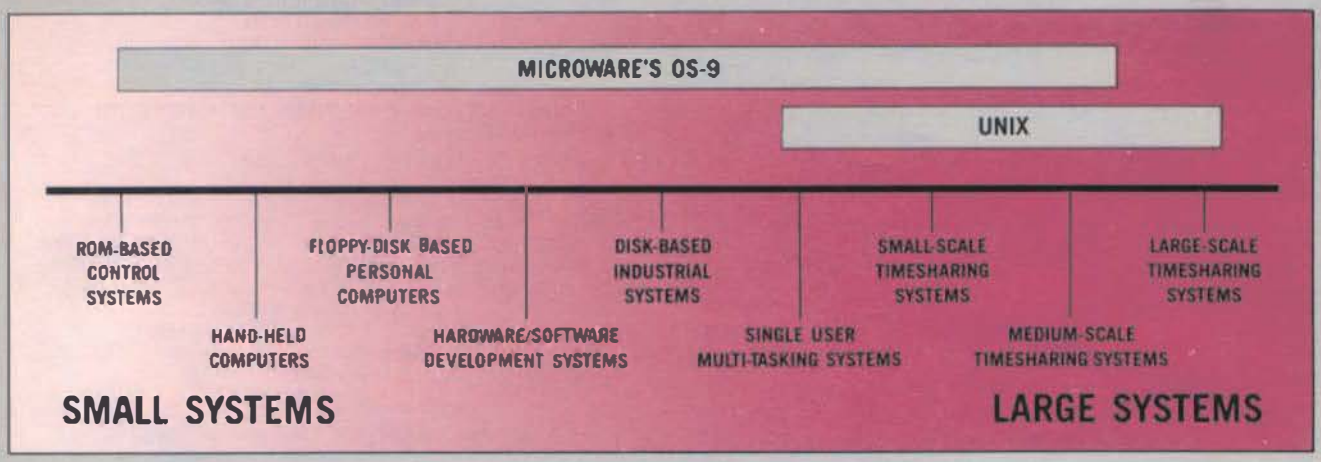
Up to Three Terminals - Instant expansion

*Trademark of Technical Systems Consultants

SWTPC

SOUTHWEST TECHNICAL PRODUCTS CORPORATION
219 W. RHAPSODY
SAN ANTONIO, TEXAS 78216 (512) 344-0241

Only Microware's OS-9 Operating System Covers the Entire 68000 Spectrum



Is complicated software and expensive hardware keeping you back from Unix? Look into OS-9, the operating system from Microware that gives 68000 systems a Unix-style environment with much less overhead and complexity.

OS-9 is versatile, inexpensive, and delivers outstanding performance on any size system. The OS-9 executive is much smaller and far more efficient than Unix because it's written in fast, compact assembly language, making it ideal for critical real-time applications. OS-9 can run on a broad range of 8 to 32 bit systems based on the 68000 or 6809 family MPUs from ROM-based industrial controllers up to large multiuser systems.

OS-9'S OUTSTANDING C COMPILER IS YOUR BRIDGE TO UNIX

Microware's C compiler technology is another OS-9 advantage. The compiler produces extremely fast, compact, and ROMable code. You can easily develop and port system or application software back and forth to standard Unix systems. Cross-compiler versions for VAX and PDP-11 make coordinated Unix/OS-9 software development a pleasure.

SUPPORT FOR MODULAR SOFTWARE — AN OS-9 EXCLUSIVE

Comprehensive support for modular software puts OS-9 a generation ahead of other operating systems. It multiplies programmer productivity and memory efficiency. Application software can be built from individually testable software modules including standard "library" modules. The modular structure lets you customize and reconfigure OS-9 for specific hardware easily and quickly.

A SYSTEM WITH A PROVEN TRACK RECORD

Once an underground classic, OS-9 is now a solid hit. Since 1980 OS-9 has been ported to over a hundred 6809 and 68000 systems under license to some of the biggest names in the business. OS-9 has been imbedded in numerous consumer, industrial, and OEM products, and is supported by many independent software suppliers.

Key OS-9 Features At A Glance

- Compact (16K) ROMable executive written in assembly language
- User "shell" and complete utility set written in C
- C-source code level compatibility with Unix
- Full Multitasking/multiuser capabilities
- Modular design - extremely easy to adapt, modify, or expand
- Unix-type tree structured file system
- Rugged "crash-proof" file structure with record locking
- Works well with floppy disk or ROM-based systems
- Uses hardware or software memory management
- High performance C, Pascal, Basic and Cobol compilers

microware® OS-9™

MICROWARE SYSTEMS CORPORATION
1866 NW 114th Street
Des Moines, Iowa 50322
Phone 515-224-1929
Telex 910-520-2535

Microware Japan, Ltd
3-8-9 Baraki, Ichikawa City
Chiba 272-01, Japan
Phone 0473(28)4493
Telex 299-3122

'68' MICRO JOURNAL

Portions of the text for 68 MICRO JOURNAL was prepared using the following furnished hard/software.

COMPUTERS-HARDWARE
Southwest Technical Products
219 W. Rhapsody
San Antonio, Texas 78216
S09-5/8 DMF disk-CDS1-8212W-Sprint 3 Printer

GIMIX Inc.
1337 West 37th Place
Chicago, IL 60609
Super Mainframe-OS9-FLEX-Assorted Hardware

EDITORS-WORD PROCESSORS
Technical Systems Consultants, Inc.
111 Providence Road
Chapel Hill, NC 27514
FLEX-Editor-Processor

Great Plains Computer Company, Inc.
PO Box 916
Idaho Falls, ID 83401
STYLO-Mail Merge

CONTENTS

Vol.VI,Issue XI November 84

Send All Correspondence To:

Computer Publishing Center
68 MICRO JOURNAL
5900 Cassandra Smith
PO Box 849
Hixson, TN 37343
Phone 615/842-4600 TELEX 558 414-PYT BTH

68' Micro Journal is published 12 times a year by Computer Publishing Inc. Second Class Postage Paid ISSN 0194-5025 at Hixson, Tenn. and additional entries. Postmaster: send Form 3579 to 68' Micro Journal, PO Box 849, Hixson, Tennessee.

SUBSCRIPTION RATES
USA
1-Year $24.50 2-Years $42.50 3-Years $64.50
FOREIGN
See Page 60

Items Submitted for Publication

Articles submitted for publication should be accompanied by the authors **full name, address, date and telephone number.** It is preferred that articles be submitted on either 5 or 8 inch diskette in TSC Editor format or STYLO format. All diskettes will be returned.

The following TSC Text Processor commands ONLY should be used (due to our proportional processor): .sp space, .pp paragraph, .fi fill and .nf no fill. Also please do not format within the text with multiple spaces. The rest we will enter at time of editing.

STYLO commands are all acceptable except the .pg page command, we print edited text files in continous text.

All articles submitted on diskettes should be in TSC FLEX™ format, either FLEX2 6800, or FLEX9 6809 any version.

If articles are submitted on paper they should be on white 8X11 bond or better grade paper. No hand written articles (hand written or drawn art accepted). All paper submitted articles will be photo reproduced. This requires that they be typed or produced with a dark ribbon (no blue), single spaced and type font no smaller than 'elite' or 12 pitch. Typed text should be approximately 7 inches wide (will be reduced to column width of 3 1/2 inches). **Please use a dark ribbon!**

All letters to the editor should also comply with the above and bear a signature. Letters of 'gripes' as well as 'praise' are solicited. We attempt to publish all letters to the editor verbatim, however, we reserve the right to reject any submission for lack of 'good taste'. We reserve the right to define what constitutes 'good taste'.

Advertising: Commercial advertisers please contact the 68 Micro Journal advertising department for current rate sheet and requirements.

Classified: All classified must be non-commercial. Maximum 20 words per classified ad. Those consisting of more than 20 words should be figured at .35 cents per word. 20 words or less $7.50 minimum, one time, paid in advance. No classified ads accepted by telephone.

GIMIX HAS THE 6809 SYSTEM TO SUIT YOUR NEEDS

HARDWARE

All systems feature the **GIMIX CLASSY CHASSIS**; with a ferro-resonant constant voltage power supply, gold plated bus connectors, and plenty of capacity for future expansion.

Static **RAM** and double-density **DMA** floppy disk controllers are used exclusively in all systems.

All systems are guaranteed for 2 MHz operation and include complete hardware and software documentation, necessary cables, filler plates, etc.

Systems are assembled using burned-in and tested boards, and all disk drives are tested and aligned by **GIMIX**.

You can add additional components to any system when ordering, or expand it in the future by adding **RAM, I/O**, etc.

GIMIX lets you choose from a wide variety of options to customize your system to your needs.

SOFTWARE

All **OS-9/FLEX** systems allow you to software select either operating system.

Also included is the **GMXBUG** monitor and, in systems with 128K or more of RAM, **GMX-VDISK** for **FLEX**.

All **GIMIX OS-9** systems include **Microware's Editor, Assembler, Debugger, Basic09,** and **Runb**; and the GMX versions of **RMS** and **DO** for **OS-9**.

All **GIMIX** versions of **OS-9** can read and write RS color computer format OS-9 disks, as well as the Microware/GIMIX standard format.

New and exclusive with **OS-9 GMX III** systems is the **GMX OS-9 Support ROM**, a monitor for **OS-9** that includes memory diagnostics and allows the system to boot directly from either hard disk or floppy.

A wide variety of languages and other software is available for use with either **OS-9** or **FLEX**.

OS-9 GMX III/FLEX SYSTEMS (#79)

The **#79** super system now includes (in addition to the above): the **GMX 6809 CPU III, a 256K CMOS Static RAM Board (#72), and a 3-port Intelligent Serial I/O Processor (#11).**

The **GMX 6809 CPU III** can perform high-speed DMA transfers from memory to memory and uses memory attributes and illegal instruction trapping to protect the system and users from program crashes. If a user program crashes, only that user is affected; other users are unaware of the problem.

The 3-Port Intelligent Serial I/O Board (#11) significantly reduces system overhead by handling routine I/O functions; freeing the host CPU for running user programs. This improves overall system performance and allows user terminals to be run at up to 19.2K baud.

with dual 40 track DSDD drives	$5998.79
with dual 80 track DSDD drives	$6198.79
with **#88** dual 8" DSDD drive system	$7698.79
with **#90** 19MB Winchester subsystem and one 80 track	$8898.79
with a 47MB Winchester subsystem and one 80 track	$10,898.79
with a 47MB plus a 6MB removable pack Winchester subsystem and one 80 track drive	$12,398.79

TO ORDER BY MAIL: SEND CHECK OR MONEY ORDER OR USE YOUR VISA OR MASTER CHARGE. Please allow 3 weeks for personal checks to clear. U.S. orders add $5 handling if order is under $600.00. Foreign orders add $10 handling if order is under $200.00. Foreign orders over $200.00 will be shipped via Emery Air Freight COLLECT, and we will charge no handling. All orders must be prepaid in U.S. funds. Please note that foreign checks have been taking about 8 weeks for collection so we would advise wiring money, or checks drawn on a bank account in the U.S. Our bank is the Continental Illinois National Bank of Chicago, 231 S. LaSalle Street, Chicago, IL 60693, account #73-32033.

OS-9 GMX I / FLEX SYSTEMS #49

The **#49** systems include 64KB static RAM, #05 CPU, #43 2 port serial board.

with dual 40 track DSDD drives	$3998.49
with dual 80 track DSDD drives	$4198.49
with **#88** dual 8" DSDD drive system	$5698.49
with **#90** 19MB Winchester subsystem and one 80 track	$6898.49

OS-9 GMX II / FLEX SYSTEMS #39

The **#39** systems include 128KB static RAM, #05 CPU, #43 2 port serial board.

with dual 40 track DSDD drives	$4498.39
with dual 80 track DSDD drives	$4698.39
with **#88** dual 8" DSDD drive system	$6198.39
with **#90** 19MB Winchester subsystem and one 80 track	$7398.39

GIMIX DOES NOT GUARANTEE PERFORMANCE OF ANY GIMIX SYSTEMS, BOARDS OR SOFTWARE WHEN USED WITH OTHER MANUFACTURERS PRODUCT.

EXPORT MODELS: ADD $30 FOR 50Hz. POWER SUPPLIES.

GIMIX, Inc. reserves the right to change pricing, terms, and products specifications at any time without further notice.

ALL PRICES ARE F.O.B. CHICAGO

Contact GIMIX for price and availability of UniFLEX and UniFLEX GMXIII Systems.

NOTE on all drive systems: Dual 40 track drives have about 700KB of formatted capacity; dual 80's about 1,400KB; dual 8" about 2,000KB. The formatted capacity of hard disks is about 80% of the total capacity.

Want to expand your system to a megabyte of Static RAM and 15 users?

Simply add additional memory and I/O boards. Your **GIMIX** system can grow with your needs. Contact us for a complete list of available boards and options.

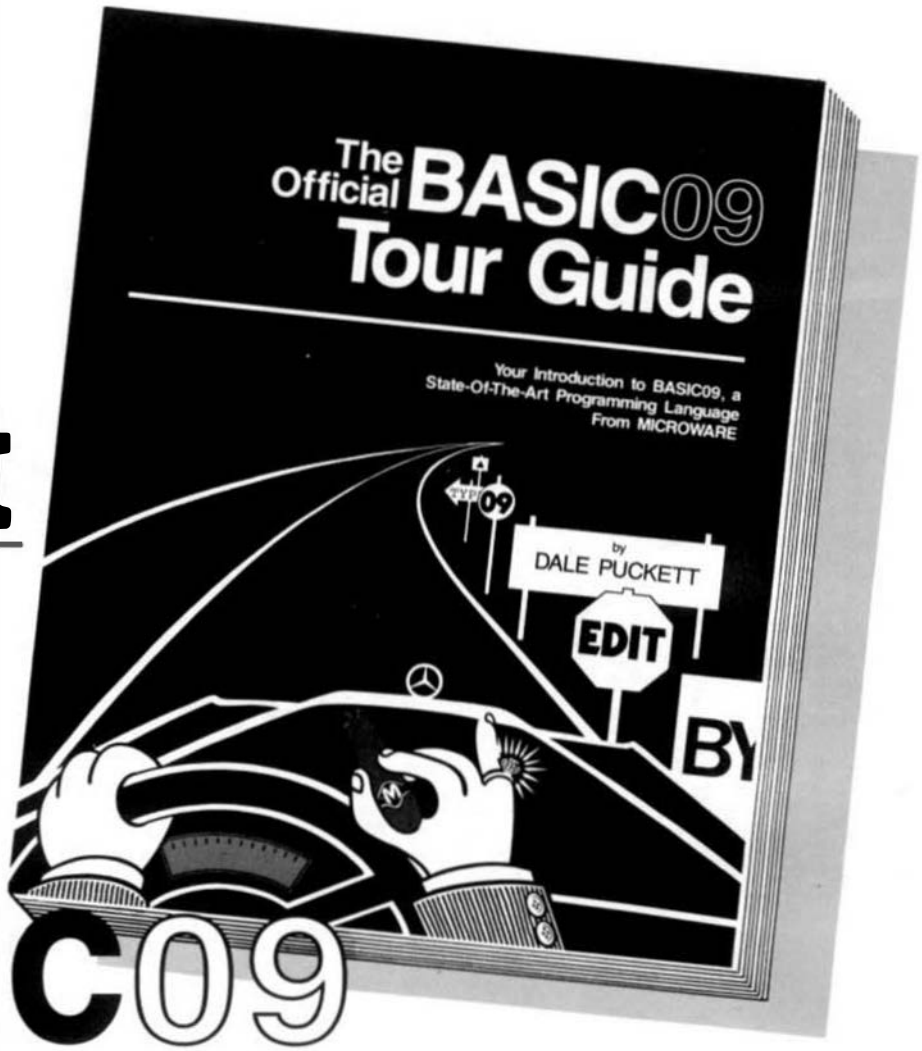
#72 256KB CMOS STATIC RAM board with battery back up	$1898.72
#64 64KB CMOS STATIC RAM board with battery back up	$628.64
#67 64KB STATIC RAM board	$478.67
#11 3 port intelligent serial I/O board	$498.11
#43 2 port serial I/O board	$128.43
#42 2 port parallel I/O board	$88.42
#95 cable sets (1 needed per port), specify board	$24.95

TRADE UP YOUR CoCo!

GIMIX will allow you up to $1100.00 credit toward the purchase of any GIMIX system when you trade-in your working Color Computer, peripherals, and original software. The trade-in value is limited to 110% of the RADIO SHACK™ list price at the time your order is placed. You pay the freight. This offer is good only in the Continental U.S.; is limited to the first 100 orders; and expires on 9/30/84. Only one trade-in per customer.

GIMIX INC.
1337 WEST 37th PLACE
CHICAGO, ILLINOIS 60609
(312) 927-5510 • TWX 910-221-4055

Get the most out of BASIC09



The **OFFICIAL BASIC09 TOUR GUIDE** is skillfully written in a friendly and easy-to-read style. Just perfect for those new to computers and to BASIC09. It's also a *valuable reference book* for programmers, engineers, students and hobbyists, providing an in-depth look at BASIC09 plus an overview of the OS-9 operating system. Comprehensive reference sections on BASIC09 and OS-9 commands are also included.

The book "maps" out your route through the Mercedes of Basics . . . BASIC09 and puts you in the driver's seat in no time. Fasten your seatbelt, sit back and enjoy the ride to perfecting your programming skills.

MICROWARE . . .

The **OFFICIAL BASIC09 TOUR GUIDE** comes from the people who wrote BASIC09. As the leader in 6809 system software, we at MICROWARE care about our users and want to help you get the most from our products.

It's Easy to Order.

Phone orders are accepted from MasterCard or VISA cardholders or for COD shipment. You can also order by mail using the coupon below. Quantity discounts are available to educational organizations and dealers. For further information contact Microware.

microware®

Specialists in system software for 68-family microprocessors since 1977.

OS-9 and BASIC09 are trademarks of Microware and Motorola.

Microware Systems Corporation
1866 N.W. 114th Street
Des Moines, Iowa 50322
Telephone 515/224-1929
Telex 910-520-2535

Please send _____ copies of the **Basic09 Tour Guide book** at $18.95 each. Add $2.00 for UPS shipping in the U.S. or $5.00 for overseas air mail per book. Iowa residents add 4% sales tax.

Name _____

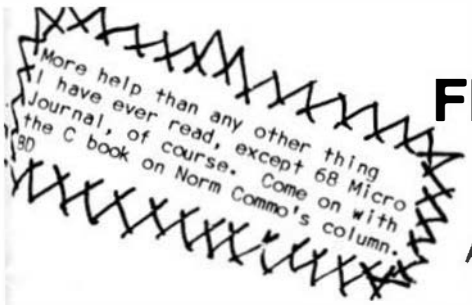
Address _____

City _____

State _____ Zip _____

☐ I have enclosed a check

☐ Charge to my bank card:
MasterCard ☐ VISA ☐
Card Number _____
Expiration _____



FLEX™ USER NOTES
THE 6800-6809 BOOK

By: Ronald W. Anderson
As published in 68 MICRO JOURNAL™

The publishers of 68 MICRO JOURNAL are proud to announce the publication of Ron Anderson's **FLEX USER NOTES,** in book form. This popular monthly column has been a regular feature in 68 MICRO JOURNAL SINCE 1979. It has earned the respect of thousands of 68 MICRO JOURNAL readers over the years. In fact, Ron's column has been described as the 'Bible' for 68XX users, by some of the world's leading microprocessor professionals. Now all his columns are being published, in whole, as the most needed and popular 68XX book available. Over the years Ron's column has been one of the most popular in 68 MICRO JOURNAL. And of course 68 MICRO JOURNAL is the most popular 68XX magazine published.

As a SPECIAL BONUS all the source listing in the book will be available on disk for the low price of: FLEX™ format only — 5" $12.95 — 8" $16.95 plus $2.50 shipping and handling, if ordered with the book. If ordered separately the price of the disks will be: 5" $17.95 — 8" $19.95 plus $2.50 shipping and handling.

Listed below are a few of the TEXT files included in the book and on diskette.

All TEXT files in the book are on the disks.

LOGO.C1	File load program to offset memory — ASM PIC
MEMOVE.C1	Memory move program — ASM PIC
DUMP.C1	Printer dump program — uses LOGO — ASM PIC
SUBTEST.C1	Simulation of 6800 code to 6809, show differences — ASM
TERMEM.C2	Modem input to disk (or other port input to disk) — ASM
M.C2	Output a file to modem (or another port) — ASM
PRINT.C3	Parallel (enhanced) printer driver — ASM
MODEM.C2	TTL output to CRT and modem (or other port) — ASM
SCIPKG.C1	Scientific math routines — PASCAL
U.C4	Mini-monitor, disk resident, many useful functions — ASM
PRINT.C4	Parallel printer driver, without PFLAG — ASM
SET.C5	Set printer modes — ASM
SETBAS1.C5	Set printer modes — A-BASIC
	(And many more)

**Over 30 TEXT files included in ASM (assembler) — PASCAL — PIC (position independent code) TSC BASIC-C, etc.

NOTE: .C1,.C2, etc. = Chapter 1, Chapter 2, etc.

This will be a limited run and we cannot guarantee that supplies will last long. Order now for early delivery.

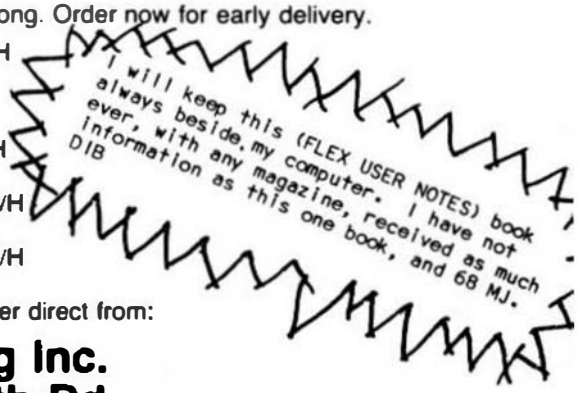
Foreign Orders Add $4.50 S/H

Softcover — Large Format

Book only: **$7.95** + $2.50 S/H

With disk: 5" **$20.90** + $2.50 S/H

With disk: 8" **$22.90** + $2.50 S/H



See your local S50 dealer/bookstore or order direct from:

Computer Publishing Inc.
5900 Cassandra Smith Rd.
Hixson, TN 37343
(615) 842-4601

TELEX 558 414 PVT BTH

™FLEX is a trademark of Technical Systems Consultants

$4,325
FOR A WORLD-CLASS SS-50 COMPUTER

Smoke Signal's VAR/68™ gives you:

- Fabled Chieftain performance that led the pack in tough Benchmark surveys
- Integrated, easy-to-use software that covers your *complete* business needs
- Proven reliability backed by our exclusive Endurance-Certification Program
- Extremely good looks and unsurpassed operator comfort

$4,325: A PRICE CALCULATED TO GET YOU HOOKED ON THIS BLOCKBUSTER SS-50

That price buys you a VAR/68 computer with multi-user, multi-tasking capabilities, and an ergonomically designed terminal. You get 128K RAM—expandable to 1mb. Eight serial ports, up to 16 if desired. Two parallel ports—and more are available. Plus a long list of other impressive capabilities.

Smoke Signal's experience allows us to offer OS-9 and other UNIX-like, and multi-user operating systems.

The styling is completely new—fashioned for the utmost in operator comfort. And it's remarkably compact. VAR/68 is a combination of great performance and good looks demanded by the office of today.

VAR/68 IS TOUGH, BUT SMOKE SIGNAL GIVES YOU *EXTRA* PROTECTION

(1) Your new computer is Endurance-Certified before delivery. That's an exclusive quality-assurance process that guarantees perfect operations from day one. (2) Our Advance-Replacement program is yours for a low fixed charge. (3) You get instant diagnostic service by telephone. It's free. (4) Normal repairs are handled with super speed. (5) Software *and* hardware support are part of doing business with Smoke Signal.

TOTAL INTEGRATED SOFTWARE GIVES YOUR BUSINESS *SOLUTIONS* INSTEAD OF PROBLEMS

Powerful business application programs are ingeniously interlinked to give even untrained operators a quick, smooth upper hand. The VAR/68 is a joy for first-time users, and an unprecedented productivity tool for *anyone* who wants new dimensions of control over critical business matters.

This screen tells part of the story:

GET A BIG DISCOUNT ON YOUR INITIAL ORDER

Most re-sellers can save up to 42 percent—even on small orders. Smoke Signal's price schedule is a powerful profit-maker for dealers of almost every description.

CALL SMOKE SIGNAL OR WRITE FOR MORE INFORMATION ON THE VAR/68 COMPUTER FAMILY

VAR/68™ BY SMOKE SIGNAL

The computer designed for your specialized application software

SMOKE SIGNAL

Products and Support for VARs

31336 Via Colinas • Westlake Village, CA 91362-3984 • (818) 889-9340

THE 68000 FROM SMOKE SIGNAL!

ADD 68000 AND UNIX™ * TO YOUR EXISTING SS-50 COMPUTER AT PRICES 50% TO 75% OFF LIST

THANK YOU

Seven years ago, Smoke Signal was founded to sell state-of-the-art computer products, by mail, to individual professional programmers and hardware engineers. At that time, most big companies did not believe in the power or future of microcomputers for serious computing applications. Only after you, the individual computer user, proved the viability of the microcomputer was Smoke Signal able to sell systems for business uses. However, as we progressed to become the leader in SS-50 systems, we had to add the sales and technical support services demanded by these business customers — and our prices for complete systems reflected these added costs.

With the introduction of our 68000 products, we wanted to find a way to say thanks to you, our original customers, the individual computer users, and still offer complete sales and technical support to our business customers for complete systems. We think this offer accomplishes both of these goals. We are offering you a choice of upgrade kits that will bring any SS-50 computer up to the electrical equivalent of our complete 68000 computer systems at prices far below complete system prices. In fact, the prices offered are 50% or more off our normally low prices for the components contained in the upgrade kits.

This special offer is limited to one upgrade kit per customer and is our way of saying thanks to those of you who had confidence in us from the beginning.

THE UPGRADES

The following upgrade kits were designed so that any SS-50 system can be upgraded to 68000/UNIX.

SWTP UPGRADE..... $2,800.00
Contains: LMB-1A SS-50C Motherboard, DCB-4A floppy controller, PSA-1 Winchester/Tape DMA interface, SCB-68K 68000 CPU, SER-2 dual serial board, 5Mb Winchester and controller, power supply, all cables, and REGULUS.

GIMIX UPGRADE..... $2,500.00
Contains: Same as SWTP Upgrade except allows you to use your GIMIX motherboard, serial board and Winchester power supply.

Users of standard SMOKE SIGNAL systems may choose one of the following upgrade kits:

For SSB floppy based systems:

SS-FD UPGRADE $2,100.00
Contains: SCB-68K 68008 CPU, PSA-1 Winchester/Tape DMA interface, 5Mb Winchester and controller, power supply, all cables, and REGULUS.

For SSB Winchester based systems:

SS-HD UPGRADE..... $500.00
Contains: SCB-68K 68008 CPU and REGULUS.

COMPLETE SYSTEMS

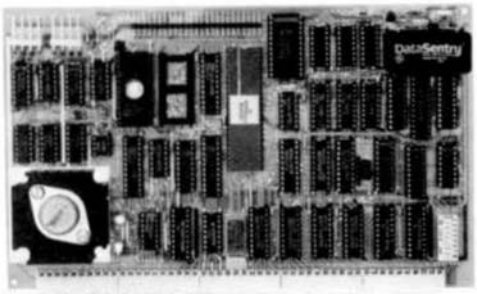
SMOKE SIGNAL is also making available complete VAR/68K™ systems at dramatic discounts. This offer is only available through SMOKE SIGNAL dealers. Contact SMOKE SIGNAL directly for information about how to order a complete VAR/68K system.

RULES OF THE OFFER

1) Limit, one upgrade system per customer.
2) Prices valid through December 31, 1984.
3) Orders must be accompanied by full payment in the form of individual check or credit card authorization.
4) Support will only be provided for systems containing the following SMOKE SIGNAL boards: SCB-68K, DCB-4A, PSA-1, and a motherboard such as the LMB-1A with extended addressing and main terminal I/O at FF7E8.
5) While we feel that most static RAM boards will work with these upgrades, we only guaranty compatibility with systems containing SMOKE SIGNAL static or dynamic R M.

VAR/68K is a trademark of Smoke Signal.
REGULUS is a registered trademark of Alcyon Corp.; UNIX is a registered trademark of Bell Laboratories; OS9 and OS9/68K are trademarks of Microware; MACSBUG is a trademark of Motorola Inc.

***Regulus the OS offered is UNIX Compatible**



PRODUCTS

The heart of all these upgrade kits is SMOKE SIGNAL'S new SCB-68K 8 MHz 68008 CPU Board. This standard (5½" x 9") board will replace a SCB-69 CPU Board in any SMOKE SIGNAL computer with current revision boards. This board contains a real-time clock with battery back-up, 2 EPROM slots for up to 64K bytes of storage, a MACSBUG™ type monitor along with an auto boot loader and a mnemonic disassembler, plus many more features.

All upgrades also come standard with REGULUS™, a UNIX like operating system which is totally compatible with UNIX. REGULUS supports real-time tasks, shared memory, record locking and contains a shell similar to the Berkeley C shell. Along with the operating system, you get C, an editor, assembler, linking loader, interactive debugger and a word processor.

SMOKE SIGNAL is also including in many of the kits the DCB-4A double density floppy controller which can handle up to four 5" and four 8" floppies and contains 1K of buffer RAM for fast disk transfers; the PSA-1 Winchester/Tape DMA interface board which has taps for SASI and Priam disk interfaces as well as a tap for 90 ips tape streamers which are supported under both REGULUS and OS9™; either a M-256-X or M-512-X dynamic RAM board with over two years of field proven reliability; and the LMB-1A heavy duty motherboard with gold plated connectors, extended addressing and on-board baud rate generator with ten selectable baud rates.

SOFTWARE

Software and Software Support is available only from Smoke Signal dealers. Spread Sheet, Word-Processing, Relational Database, C, Basic and Cobol are all available now. Additional system's software is becoming available every day because of the UNIX compatibility.

SMOKE SIGNAL dealers are also offering Microware's OS9/68K™ to purchasers of these upgrade kits. SMOKE SIGNAL will offer other Microware 68000 products as they become available.

SUPPORT

Even at these "lower than PC" prices, we're not going to leave you with "PC" type support. We've arranged with one of our very technically qualified dealers to provide you with add-on software and technical support. In addition to answering your questions on how to convert your system to the 68000, he has a group of his customers who are themselves computer experts who are joining in a network that will help with even the most technical questions. We hope you will contribute your ideas to the network so that we all can benefit from new and fresh thinking. Complete details of the support available are included with the upgrade systems.

ORDER FORM

Fill in your name, address and phone number below. Your order will be shipped UPS so please do not use P.O. Box. Check items being ordered on form. Add prices for all items selected. CA residents must add 6% for sales tax. Total the amount for your order and check payment method below.

Name _____
Address _____
City, State, Zip _____
_____
Phone _____

Payment: ☐ Enclosed Check
☐ VISA
Mastercard
Card # _____
Exp. Date _____
Signature _____

☐ SS-FD UPGRADE	$2100	_____
☐ SS-HD UPGRADE	500	_____
☐ SWTP UPGRADE	2800	_____
☐ GIMIX UPGRADE	2500	_____
☐ M-256X RAM	648	_____
☐ M-512-X RAM	948	_____
☐ SER-2 I/O	85	_____
☐ 20Mb H RD DISK (instead of 5Mb)	800	_____
Sub Total		_____
CA residents add 6%		_____
Total		_____

SEND COMPLETED ORDER FORM TO:
SMOKE SIGNAL
31336 Via Colinas, Westlake Village, CA 91362-3864
(818) 889-9340

Flex User Notes

Ronald W. Anderson
3540 Sturbridge Court
Ann Arbor, MI 48105

Label It

From time to time, some of you send me a bit of software on a disk. Recently, I've received a disk or two with NO LABEL. I give fair warning and notice here. A disk with no label in my scheme of things is considered to be blank or at least ready to be reformatted. When I scrap a disk, I pull off the label and put it in the pile. When I need a disk, I find one without a label and format it. If you send me something and you really want me to look at it, LABEL it, and I don't mean by writing something in the quarter square inch blank area of the disk manufacturer's label. While you are at labeling the disk, please include complete information about the disk. Generally it would be best to send software in single sided, single density (40 track is OK) format. I can read that with no trouble. Any format that is more dense, is supplier dependent. That is, a double density disk formatted with SWTPc versions of flex can't be read with Gimix FLEX and vice versa. I recently received a disk with minimal label, and found out by reading the system information record, that it had 80 tracks of ten sectors each. That makes it single sided, single density, but 80 tracks. I can read the disk at work, and copy it to another format, but it would be nice if the label would give me a clue as to the format!

If you are like me, you run out of the disk labels supplied with a box of disks, long before the disks are "dedicated" to some permanent files, or are worn out. A long time ago I introduced another FLEX user to Avery removable labels, available in nearly any office supply store. Their type S-1648 are 1" by 3" and are ideal for labelling 5 1/4" disks. Their S-6424 are 1 1/2" by 4" and are ideal for labeling 8" disks. The "removable" labels come in yellow boxes. They are easily removable for a year or two, and come off with a little greater difficulty after that. The non-removable ones come in blue boxes, and are removable only with solvent, even immediately after application. If you re-use disks as I do, you will want the removable kind. They won't fall off, but they can be peeled. Incidentally, stubborn sticky old labels can be removed with some care, by applying a small amount of paint THINNER (NOT PAINT REMOVER) to the label and letting the adhesive soften. After peeling the paper away, a little paint thinner on a paper towel will remove the remaining adhesive. Be careful not to allow the thinner to flow into the disk jacket. I hesitate to mention that lighter fluid (naptha) will work even better. I use it safely because I don't smoke. If you do use it BE CAREFUL.

History

A couple of days ago I received a copy of a review by Don Williams of the book "Fire in the Valley", a history of the microcomputer industry, and a very inaccurate one according to Don. It seems that the authors of this book had gotten all fired up over the early 8080 and Z-80 machines, and nearly forgot that those other processors existed at all. In fact, Southwest Technical Products Corp., was the very first microcomputer company, and they, unlike many of the original S-100 suppliers, are still around. Don's review should have appeared by now, so I won't repeat what he said. You can look it up for yourself.

Don mentioned names of several people who were instrumental in the early days of 68XX computers, both in the hardware and software areas. I'd like to expand a little on the software end of things. Those of us who started early on this bus will remember Bob Ulterwyk as the author of a very good BASIC interpreter (which cost $15 in the cassette version, from SWTPc). I'd like to think a little about some of the other names in the software area, folks who contributed to the wealth of software that we have, and some who are now contributing.

Don didn't mention the late Ed Smith, who was active in the area of assemblers. Ed had a relocatable assembler available, and if I remember correctly, also a disassembler that he called a "Source Generator". Don did mention Technical Systems Consultants. I remember their early ads in Kilobaud Microcomputing during its first year. TSC was selling games in 6502 and 6800 assembler code. Soon after that, they had an assembler, EDIT and PR, a rather complete text processor, (all on cassette initially). Later when I added a pair of disk drives, the operating system was a thing called FDOS by Bob Ulterwyk again. When FLEX2 came along, of course, TSC modified what they then had available so that it would work from and to disk. Tom Crosley was responsible for a very nice 6800 language called SPL/M, and the ever elusive PIE text editor (still not ever distributed by anyone for 68 and 6809).

It was not until about the time of the coming of the 6809 that software really "took off", however. At about that point a company called Lucidata came on the scene with what I called at the time "The first non-toy compiler for the 6800. I suppose some folks would argue with me that both Strubal+ by Jack Hemenway and Bob Grapel, and *Software Dynamics BASIC compiler by Ira Baxter, predated Lucidata Pascal, and I would have to yield to the point that they were earlier. Lucidata Pascal was written by Nigel Bennee and Dave Gibby with able help from Eileen Bennee. It was followed shortly by a 6809 version, and Lucidata is still alive and well in England. *Eds Note: Software Dynamics and Ira Baxter are also still around, I hear from Ira every now and then. DMW

Just about that time, the software started appearing more rapidly. Stylograph by Bob Bundy appeared (Bob has sold the rights to Stylograph to Great Plains Computers since then). Tom Crosley sent me a copy of PIE for evaluation, with intentions of marketing it for the 68XX community. In fact, if you have all the issues of '68' Micro Journal, you will find reviews of PIE by myself and another 68XX enthusiast named Randy Lewis.

At about that point, there was a sudden increase in the availability of standard languages. Omegasoft Pascal by Bob Reimiller appeared, and the company with which I was associated at the time, bought a copy and used it for a large software project. Some of the early "C" compilers began to appear at that time also, some of which have apparently disappeared from the marketplace. Currently, we have the efforts of James McCosh in his various versions of a "C" compiler for FLEX, Uniflex, and OS-9. I'll have to plead lack of knowledge of programmers names on some of these, but we have the Introl "C" compiler, Compusense Crunch Cobol, the very new K-BASIC compiler by Frank Hoffman, and of course PL/9 by Graham Trof .

I should mention that Frank Hoffman has written a very nice line of cross assemblers that run on the 6809, to produce code for nearly all the other 8 bit processors and some of the 16 bit, at least the 68000 at this point. Who have I left out? Well, there's Al Jost of DynaSoft, who wrote DynaSoft Pascal, Dyna-C, DynaStar editor, and DynaForm text processor. Then there's Phil Lucido who was responsible for Dynamite, the disassembler, and of course Bud Pass with his Super Sleuth, Tabula Rasa, and a number of other software items.

Oh, and there are the implementations of Forth for the 68XX by Ray Talbot, and the slight variation X-FORTH by Chuck Eakers. How about John Alford's Screditor III also. Let's see. I have to mention Peter Stark, (Star Kits) who had the first spelling checker for our systems. His other products include HUMBUG, the monitor, and STARDOS, a disk operating system that is pretty much like FLEX. Speaking of spelling checkers reminds me that Dale Puckett has contributed to our software too with his spelling checker, a neat program that tested the "level" of english text, and some other goodies. Almost forgot Dan Farnsworth who wrote SPELLB and is presently working on several other products.

I hope I haven't left too many people out. Unfortunately, I haven't gotten acquainted with all the people at the companies that are larger. I can't name the people at TSC (other than Dave Shirk and Dan Vanada) but I should mention that in addition to FLEX and their early EDIT, PR, and 68 assembler, they were responsible for Extended BASIC, DEBUG, a Sort Merge package, a Pascal, two 6809 assemblers, and a number of other software products.

At Microwere, I can only name Ken Kaplan. Of course you all know that Microware wrote the OS-9 operating system and BASIC09. They are also responsible for some other software that we have used from time to time such as A/BASIC. I knew some of the folks at Great Plains about the time that they started, but we have lost touch.

Well, that's enough of that. Do you get the idea that a lot of people have been involved with these 68XX systems? I agree with Don that some of them should have

deserved mention in the history of the microcomputer industry, along with the hardware suppliers, SWTPc, Smoke Signal, and GIMIX.

I'm with you Don, regarding the longevity of our SS50 systems. I just bought a used one that was 6 months older than my personal one, having been purchased in the Fall of 1976 from SWTPc. It has been upgraded and modernized over the years, and all it needs is a DMA disk controller and a pair of 8" drives to make it a very capable development system, the equal of mine and the more modern SWTPc system we have at the company (bought used also). Actually it won't be used that way, however. I have added the drives and disk controller to a development system at work so we can store customer software for each machine that we deliver, on a 5 inch disk (one disk for each machine). the remainder of the system will have a newer disk controller and a pair of double sided drives added to it to make another system.

With the addition of a 256K Computer Excellence memory board to my personal system, I have just upgraded to 2 Mhz operation AND added the capability of 752 sectors of Virtual Disk as well. These two upgrades make the chore of compiling programs much more pleasant. I can generally put the compiler on Virtual Disk and decrease the compile time by about half over the 8" disk drives. The speed change also makes a significant difference in the compile time. The company for which I work "most of the time" has put probably 150 6809 stand alone computer systems into the industrial applications market in the past couple of years. We went SS-50 bus because of the availability of off-the-shelf cards for our initial development of hardware. Though we have since designed our own hardware cards (after we figured out what we needed), we are still on the SS-50 bus, which has several advantages for us. First of all, we can plug any of our cards into the SS-50 development system for testing and debugging. Second, if we need something we don't have, we can generally buy it on the SS-50 bus.

I think I've probably said more than enough in defense of the SS-50 bus. Don, I couldn't agree more, that we who use this bus and the 68XX processor are doing significant things with our systems and hardware.

Peter Stark Comes to Ann Arbor

No kidding, one day last week I received a call from Peter indicating that he and his family were about 35 miles from here, and that they were going to camp for a few days just about 10 miles from our house. After a couple of false starts due to some car problems that the Starks had, we managed to get together for an hour's chat, and we got our families together for a Saturday evening. I've talked to Peter a number of times on the phone, and we have corresponded for several years with reference to software. It was very nice to meet Peter and his family. I hope they had no further car problems on the remainder of their vacation trip. Keep in touch, Peter.

Personal Note

If Don will permit me a couple lines here, I have lost touch with the world's greatest microcomputer salesman, who, last I heard, now works directly for Motorola somewhere in the Detroit area. If any of you know where I can get in touch with Jack Halliwell, please send me a note with the information, or if by chance Jack, you still read my column, please get in touch with me at Hines Industries in Ann Arbor.

Response

In the Aug/Sept issue of '68' Larry Williams printed a letter from a reader, which for good reason was anonymous. I'd like to comment on that letter here. First of all, I have neither any association with Radio Shack nor any reason to like or dislike them strongly, so I think I can be reasonably neutral in this discussion.

First, I think the writer's premise was all wrong. No wonder everything went poorly. I quote "I owned a computer from SWTPC which included dual disk drives and the FLEX operating system. Needing a better version of a 6809 based computer I bought a Radio Shack 64K Color Computer." Now who but someone who expected something for nothing could expect a $200 Color Computer to be an upgrade from a "component system". The writer missed the whole point of component systems. Regardless of the state of his system, I would have gladly bought a Color Computer and traded him even for what he had! Let me explain further.

I bought my original SWTPC system with 2 4K memory boards a "C" interface and a 6800 processor. I also bought a cassette interface and their CT-1024 terminal kit. That was the start. Since then the upgrade process has continued for 8 years. I added a pair of disk drives a year later. I bought a few 8K memory boards, and added a couple of 16K to round out 56K of memory when I bought a pair of 8" drives, a power supply and a DMAF disk controller board. SWTPc continues to this day to release hardware modification instructions for upgrades. The 6809 pr essor board was installed with half an hour's worth of simple modifications to that old mother board. I just a couple weeks ago replaced my memory boards with a 256K dynamic board from Computer Excellence. That enab d me to upgrade the system to 2 MHz operation, since I had the 2 Mhz MP-09 board but was limited by a 32K dynamic RAM board to 1 Mhz operation previously.

You might ask how much of the original computer I still have, and how much it has all cost in the long run. Of course all that is left of the original is the box and mother board. Have you checked the Ad's for boxes and mother boards? They currently cost more than a Color Computer! The point is not how much I have spent, but what I have been able to save over a long period by not having to throw the whole system away and buy a bigger one every couple of years. I don't think my investment per year has been all that much. The point is that with a Color Computer you have a device with very limited expansion capability. (I know, there are 80 column display boards, lower case boards, and even bus expansion systems that allow you to add on ad nauseum, but you still have the original system with some fixed built in limitations.

The original SWTPc system had a parallel port programmed as a serial interface, more or less like the CoCo has now. It was limited to 1200 baud maximum, which seemed fast at the time, but is intolerable with a screen editor. SWTPc recognized the limitations about the time FLEX2 for the 6800 appeared, and indicated that FLEX wouldn't run with the old "C" interface but that a true serial interface was required. Cost of that interface was, I believe, $40. Problem solved. Now the user could have interrupt input handling and a screen editor could go about busily updating the whole screen and never miss a character being input by a 100 WPM typist. I don't type that fast, but I think I hit or exceed that speed in bursts now and then.

The CoCo has a parallel port similar to the old SWTPc arrangement. You can't change it except to buy one of the bus expander boxes for more than the cost of the CoCo, and then you still have to buy the serial interface for it! I have a CoCo. I really gave it a try for some serious use. I can't edit on it with my external terminal, because it misses characters. Anyone used to an 80 column terminal will NEVER be happy with 32 or 40 or 51 for any SERIOUS use.

I'm going to get my two cents in here about operating systems too. OS-9 is a very good one. On a system with four terminals and a hard disk, it is right at home, looking very much like UNIX. Putting it in a CoCo is like putting a 1000 HP jet engine on a rowboat! OS-9 has a "software clock". Due to the CoCo using interrupts for its disk accesses, the time update ceases whenever you access the disk. The clock loses time quickly. You have "multi-tasking" so you can attach a real terminal to the serial port. Because of the port actually being a programmed parallel port, you can only run the external terminal at 300 baud. Any faster rate will cause missed characters, because you can't shut off the other tasks even if you only want to run the single external terminal. The clock interrupts keep coming and the system keeps looking at the CoCo keyboard as well. That shoots down using a standard terminal with the CoCo and S-9. Let me again emphasize that the problem is not with OS-9 but is a built in limitation of the CoCo.

You've probably heard about the programmed serial port vs the "true" serial port before. Maybe I can explain what the problem is. The programmed serial port inputs a character by waiting for a "start pulse" from the external serial data source. When this is received, the program enters what is called a timing loop to wait for the next serial bit to arrive. It samples that bit and again waits, etc. until it has one character stored. If the processor is doing anything but running the input character program (that is, waiting for a start pulse) when you type a key on a serial terminal, it will be "out of sync" and miss the key or interpret it incorrectly. A "true" serial port contains the hardware necessary to detect the start bit and capture and store a whole

character without any help from the processor. Further, it transfers the character to a "receive buffer" and can actually begin to accept a second character before the processor has to do anything. When the character is transferred to the receive buffer, it sets a "flag" in a control register to indicate "receive buffer full". In operation with OS-9 it also signals the IRQ interrupt line. The processor will then stop and read the character. Even without the interrupt feature, if the program that is being run checks for the RBF flag faster than the user can input characters, none are missed.

Now to get back to the main point, I'm NOT saying the CoCo is a toy or a useless pile of junk. You can't begin to put together a SS-50 system for its price. If you want to find out whether you will take to computing like a duck to water (as I did) without investing $5000, by all means get one. It can grow with you to the extent of adding disk drives, FLEX, OS-9 (if you like) a printer, an 80 column display board (via a monitor) and a lot of other things. If you really get serious about computing, you WILL outgrow the CoCo. If you don't get serious about computing, you are not out a fortune, and you can probably sell what you have to someone else, or as Lane Lester suggested in his letter in the same issue of '68' use it for a doorstop. (Or buy some game software and use it to entertain the family and guests.)

With a CoCo, if you are willing to put up with the display format, you can run BASIC, PASCAL, "C", PL/9 and several other languages. You can certainly learn to program with it.

With a SS-50 system, you have an "infinitely expandable" system that can be upgraded periodically without starting over again from scratch each time. Will I have a hard disk on my system some day? I think that is inevitable. Will I have a 68000 processor in it? That is most likely to happen too, if someone is kind enough to provide a usable operating system that is, or can be made compatible with the hardware. SWTPc has not let me down so far. All the upgrades have been possible with minimal foil cuts and jumpers. What is really inevitable in the area of microcomputers is progress. SWTPc couldn't possibly have envisioned the availability of a 256K memory board back in 1976 when they built their first machines, but it was little trick to get it to run in that old box with the original mother board in it!

The letter that started all this contained a complaint that technical information is not available from Radio Shack. I can't dispute that fact, and it is really too bad, because a fair percentage of those buying the CoCo are well able to add to it and make it more suitable for their needs given sufficient information. SWTPc has ALWAYS from day 1 provided COMPLETE documentation. That includes parts lists and schematics of EVERYTHING in the system... The earlier documentation even included the source listing of the monitor software. Though the latest SBUG-E source has not been officially released, there are many copies of well commented listings floating around among the users.

If there is any reason for criticism of SWTPc and TSC who are responsible for FLEX, it might be that they gave out too much information. FLEX is understandable, and TSC documented disk formats and file formats to the extent that it is virtually impossible for anyone to "copy protect" a disk that contains software that runs under FLEX. A knowledgeable FLEX user can "break" the protection scheme in short order. (I know, I've done it myself on a couple occasions in which suppliers have tried to protect some things, eg. FHL's PUTBOOT in their CoCo FLEX.)

On the other hand, the relative "transparency" of FLEX doesn't get in the way of programmers writing application software, and that, I believe, is why so much software exists for this small sector of the computer industry.

As usual, I've gotten too wordy and overrun my target size for this column. See you next month.

Editor's Note: The text of my 'review' of Fire in the Valley is appended hereto. It is strictly MY opinion, and **I stand by it.** So far I have received about 30 or 40 replies to a short mailing I did to advertisers and others I thought might me interested. All but **one** agreed. The lone dissenter felt that although the book was indeed biased, misleading and full of whole and half truths, it would be better off to not e acerbate the situation. Also that I was also biased in my opinion. **That is where the difference is - I tell you so - they [redacted] do!**

I guess there is merit to such argument, however, in this case the widespread circulation of this book, from a publisher who in the past has enjoyed a fairly decent reputation, makes it all the worse. To us they will always be suspect hereafter. The errors and omissions appear more deliberate than accident. A continuing example of what the S50 bus crowd has endured from other publication sources. For many of us who depend on the S50 Bus group, for a living, and personally know or knew many who were ignored, it is an **affront.** Worse even than some of the rudeness and arrogance I have experienced over this since.

We have a few who were given their start and have been accepted by YOU the S50 Bus users, your support and **$$$$$$$$'s** as well. Now other pastures look greener, soon they will wander as other have before. I wish them well; I sincerely desire that they will make it, better so than some of the others who went. The expansion and expense of doing business 'away from home' is far greater than most ever imagined in their wildest scheming.

We have tolerated far more that those others will. We require no 'national' service, we still ship bad product back to 'the factory'. We are still willing to wait a few weeks while you get around to our stuff. Oh, I know some of the others didn't like it, but who cares, they are no longer with us, they too are 'over there'. We have accepted 20 to 50 version of your software, while you worked on the 'bug' between other new projects. We paid our good money for the boards with kludges, patches and wire jumped all over the place. Some of them you never did fix, and never will, but we endured. We have become a group of 'work-arounders'! We are very unique in that we can actually fix it sometimes when you can't - or won't! Try that on those guys 'over there'.

All of which makes us the more thankful for the faithful who stick by us, help us with support and product up-grade, answer our questions without snide remarks as to our mental state, and all the other attention given our inquiry, even if we don't know quite as much about it as you do! We may not be the largest market you could service but we appreciate and support. Wait till they get 'over there'. You and I will **still** be around after some are devoured 'over there'. A pity!

Stupid as it sounds they even insist on manuals you can read, they don't like dot matrix, especially if it has pencilled in changes, slashes through, etc. They want type setting, cartoons or at least some type illustrations, they want hard binding, not looseleaf. Not that it is better, but that is what the new competition is doing. A lot of floss and flurry. Why we even support, (over 80% of us) a magazine that is photocopy and computer generated. Bet they wouldn't, that is unless it had what we (or they) wanted and was not getting elsewhere.

And marketing cost - Man! - most of the smaller entries need hundreds of thousands of dollars just to get into the fray. That was with like 6 or better zeros! No more black and white under thousand buck (and for some much less) per page of advertising. Not only that but which of the multitude of magazine do you advertise after going 'over there'. Of course you could try it as several have and not advertise - but then hunger gnaws deep, and by that time it really won't matter, the hole is dug by then - just a matter of time. And if you attempt to 'match' the better known and more popular guys (notice I did not say better (quality wise)) You had better have a direct line into the US Mint, or be willing to take on 'venture money' to the extent that you wake up some day and find you don't have an office to goto anymore. Those guys insure their best interest. And after all what have you got - other than a good (I would hope) product? The old mouse trap saying gets more true each passing day, in this business.

I guess there is a lot more I could say, but I trust that I have made my point. I wish all of you well, I sincerely do, but I also have an obligation to thousands of readers, who paid me good money, mostly hard earned, to keep them informed of stuff to buy and information on how to use it and fix it if necessary. And that includes letting them know if you intend to leave the ole bus. And are or are not going to be expected to continue to support what you took our money for.

Why? Simple, if you are not going to be around to support, fix and maintain to current standards, then we want to know. After all we all have too many dearly paid for products that, like war orphans, have been abandoned. If you sell it to me and there is absolutely, I **mean ABSOLUTELY** nothing wrong with it, and it will never need fixing, well then I guess the above won't apply to you. Else I am concerned!

If we are just a short term, stepping stone then let us know. There are plenty of others willing to stand behind and support their product. And I for one am tired of being considered second-rate. We sure were not second-rate when you started up back then, and you

sure never let me know when you took my $$$$$'s, but at least you owe me that. And we all want to know. Surprising how many of us intent to stic around!

DMW

- - -

A Short 'Book' Review!

Reference: FIRE IN THE VALLEY, a biased, inaccurate ego-trip of the beginning of the microcomputer industry, and the folks who were involved. Not all truth and not all wrong - something I would call "Vaporware".

Recently I received another computer book, "Fire in The Valley", by Paul Freiberger and Michael S wine, published by Osborne/McGraw-Hill, Berkley, California. With the exception of a few references to some of our early pioneers, both individually and as viable microcomputer manufacturers, the reader is left with the impression that the S50 Bus, 68XX segment didn't make it. This is at the least gross stupidity on the part of the authors and publisher, and indicates the attitudes of ill informed (and liking it that way) others who have apparently spent all these years, like that long legged Australian bird, with their heads down and their butts waving.

I cannot disagree with their recital of those instances concerning the computer from the days of Babbage to the birth of the IBM PC™. Who really knows it all? But I do take a strong position concerning their views of the birth of the microcomputer era! They grossly err by omission! To say they are in total error as concerns those they DID mention, would be presumptuous on my part, however, it appears that some egos were massaged while the pen toyed unmercifully with others. To ignore a small *(however, not in the beginning) but absolutely vital and valuable part of the activity revolving around the birth-pangs of microcomputers, as we know them today, is at best, poorly researched journalism in this particular publication! The S50 Bus and 68XX crowd contributed a significant portion of the creativity of those days, and extend even to today. They should have been acknowledged throughout the warp and weave of this book, to the degree they contributed!

They made slight mention of the worlds oldest existing (still in business) microcomputer manufacturer, Southwest Technical Products Corp of San Antonio, Texas. That certainly rated mention, but didn't! GIMIX is not even listed in the index, nor is SSB, Percom or many of the others, still going strong. How about their early efforts? Maybe it is because many of those responsible for this pile of mis-information were involved, to some degree or another, in the demise of some or most of the other early day microcomputer manufacturers, due to their ineptness and/or stupidity and cannot bear to believe that the S50 Bus DID survive and is going strong! Need I mention Altair, IMSAI, SOL, The Digital Group, Polymorphics, Vector Graphic and many others. All of which had many of these 'now' experts, as directive or development personnel! What a pity, I though better of McGraw-Hill. Oh, by the way, look at their failures and look at ours, some difference - I should think it would at least rate one or more pages out of 282!

To the best I can find out, not one of the authors, editors, the publisher, etc., contacted anyone directly connected with those who have some knowledge of the S50 Bus and 68XX computers, and there are MANY. What was written about us was apparently penned by one of the S100 (failure) crowd.

It was actually ALL of us, the S100, the kits, the bare boards, the S50, the Jobs, Meyers, Wozniaks, Dons, Shirks, Gates, Mauches, Kaplans, Ulterwyks, Hammonds, and all the others (my apologies to those I left out). To read this mess one is left to believe that the S50 Bus (68XX) never MADE IT. What we did was that we made better microcomputers then, and still do! Not to mention the superior software we have always had, FLEX™, OS9™, STRUBAL™, DOS68/69™, etc. There are more S50 Bus 68XX microcomputers in heavy industrial, scientific and government high level operations than ALL the original S100 Bus systems. How many 7 or 8 year old S100 systems are still running? Not many, it was a BUST! (Talk about noise.) Most S50 Bus machines are still grinding along!! Don't believe me, well, come to our office and I can prove it from our files. We (S50 - 68XX) never made it big for several reasons, but we made it, we're still here! Where is the original S100 bus? Only the blowhards left!

If, after reading 'Fire in the Valley', you believe it, then I guess you also believe that most micros were designed over coffee and pie. Betcha it's in sequel two.

-30-

DMW

*Editors Note: At the time this industry was starting, 1975-1978, more or less, the S50 and 6800 group accounted for over 45% of microcomputers. This included SWTPC, GIMIX, SSB, Sphere, Wavemate, and a few lesser know others. Also some manufacturers such as MITS (Altair) and OSI were doing both 6800 and 8080 systems.

All of this was for the most part the results of the efforts of many individuals, during all this time Motorola seemed to make less effort on behalf of the 6800 than any of the other competing manufacturers. It seems their marketing vision was a few years lagging, always. It was and is a shame for even those who made microcomputers with the 'other' chips, notably the 8008, 8080 and Z80 acknowledged that the 6800 was the better microprocessor. Another reason was the financial 'taps' some journalist had with some 8080 microprocessor manufactures, it was always difficult to get articles published in some computer magazines of the day. **That is WHY and the ONLY reason 68 Micro Journal started up!** The 6800 had the power but not the support. So, you see why I am so hot on 'support'.

I don't intend to keep kicking a dead horse, as the saying goes. As for me it really does not matter, but it is a crock for those I actually know should have been included and got the 'stick'!

DMW

OS9 USER NOTES

by **Peter Dibble**
("OS-9 Users Notes" Columnist;
'68' Micro Journal)

More Games with Directories

Last month I discussed reading from directory files. This month I'll stay with directories and add some additional tricks.

The directory formatting command at the end of this column is a useful version of the DIR command. It doesn't illustrate any ideas that weren't covered in last column, but it is a single program that is faster to use than the pipeline of programs I presented last month.

I have found that C is a good language to write quick system level programs. Of course, assembly language still has some advantages over any high-level language; not least that almost everyone with OS-9 has an assembler. A functional directory command in assembler would be just too long for one month's column, and not interesting enough to devote several months to. So the first program for this column is an integrated directory formatting command. It is written in C. It could be translated to Basic09 without too much trouble, but that would require loading Basic09 every time you want to list a directory. Sorry, people without C.

Radio Shack is selling Microware C at an impressively low price. It is a good investment.

Think of dr as a good starting point. It is easy to get it to sort its output. Adding the ability to select only files that meet certain criteria for display is harder but useful enough to be worth the effort. Working this up into a full-screen command environment is something I've been promising myself time to do ..., but I haven't yet.

You can write directories as well as read them. There are good reasons to do this. Renaming files is one reason. The rename command simply writes a new name over the old one in the directory. Deleting and creating files are other reasons to write into directory files, but RBFMan takes care of those operations. Most other things you would want to change about a file involve writing into the file descriptor sector for the file. That's just as easy as writing the directory. Easier.

There is an easy way to make C read a directory file, but there is no equivalent method for updating directory files. The combination of attributes required to write into a directory can be used from assembler, or from the lower level parts of C, but it seems Microware wanted to make it a bit tricky to mess with directories. Before I continue let me add to their implicit warning. If you are not brave and experienced don't even think of updating a directory file!

Writing on directory files is a dangerous thing to do. If you make a mistake you can loose files, or even mess up the structure of the entire disk. DON'T jump in and try programs that write to the directory on an important disk.

After making certain that your program doesn't damage the directory under normal circumstances, think about extraordinary situations. How does the program behave if the system crashes right in the middle of the change? Can trouble start if two programs try to make a change at the same time? What will a program reading the directory while you make your change see?

Another area where you can get in trouble and discover interesting new possibilities hidden in the OS-9 file structure is the possibility for having several directory entries pointing at the same file.

There is a link count in each file descriptor sector. This count will always be one in normal OS-9 systems, but the field offers a way to tell OS-9 (RBFMan) that there are two or more directory entries pointing at a file.

This trick will certainly cause OCHECK to have fits. If you link two directory files to one another (not just with the .. file name) OCHECK will loop between the two directories forever. Even if you don't get this extreme OCHECK will note that more than one file is using the clusters belonging to the file with which you're playing. I have a deadly fascination with this trick of linking to a file several times. The parts to put it together are all there, but for some reason Microware hasn't built it into OS-9 yet.

My bet is that the reason for multiple links to files remaining dormant in OS-9 is the recovery problem this feature creates. It is impossible to update the link count in the file descriptor and change the number of directory entries pointing to a file simultaneously. There is always some way to crash the system between the two operations -- pulling the plug will work.

If the link count is greater than the number of directory entries actually linked to the file, the file will eventually be left around with no directory entries pointing at it. The disk space for the file will be allocated and there will be no easy way to return them.

If the link count is smaller than the number of directory entries linked to the file the result is worse. Eventually there will be a directory entry pointing to a file that isn't there. The sectors that used to belong to the file could be part of another file or just free; in either case the result is chaos.

It looks impossible. There is trouble whether the file descriptor is updated before or after the directory. There are two solutions.

One possibility is to live with the problem. An experienced user can fuss around with the allocation map and directory entries, and repair a damaged disk. Most of the work can be automated. Computers don't crash often. Chances are they won't crash in the middle of a directory operation....

The alternative is to use "stable storage" tricks. Every time OS-9 starts up look for evidence of a crash, and every time you update a directory prepare for one. This slows directory updates, systems startup, and even disk mounts; but it prevents users from having to worry about recovery.

Neither method sounds OS-9-like. I use the "live with the problem" method. I've never had reason to regret it, but I am prepared for the worst. The "stable storage" method is interesting ... worth a brief discussion.

Here is a way to reliably update a directory:

1 Copy the entire directory including file descriptors to a special spot, with its address known to recovery routines (in a table located at some known spot).
2 Update the copy of the directory.
3 Put the address of the old directory in the same table as the address of the new one with a mark indicating that it is old.
4 Put the address of the updated directory in the directory's parent.
5 Remove the new directory from the table.
6 Delete the old directory removing it from the table.

Step 4 involves a single operation that changes the directory structure visible to the public. Until step 4 is executed no program knows about the change. After step 4 there is a consistent updated directory.

Recovery works as follows:

```
If there isn't anything in the "table"
   no recovery necessary
If there is a pointer marked "old" and no
      accompanying new directory
   delete the old directory
If there is only a new directory in the table
      delete it.
If both pointers are in the table
      continue from step 4 in the update
      procedure
```

Things fall apart again if two processes might simultaneously update the directory, or the file descriptors attached to it. If that is permitted the protocol gets complicated. Too complicated for this column.

I'm not going to try to present a program implementing stable storage this month. Just a simple program to squeeze the null entries out of a directory.

- - -

```
1 #include <stdio.h>
2 #include <ctype.h>
3 #include <modes.h>
4 #include <direct.h>
5
```

```
 6
 7 static struct dirent DirEntry;
 8 static FILE *fopen(), *dir, *disk;
 9 /*-----------------------------------------------------------*
10  *   dr                                  directory read       *
11  *   Read and format all the important fields in a            *
12  *   directory entry and the attached FDs.                    *
13  *   To allow formatting, sorting, and searching programs     *
14  *   the best access to this data it is just printed          *
15  *   without titles.                                          *
16  *   There are no options.  A directory name may be given     *
17  *   as a command line argument.  If it isn't the current     *
18  *   data directory will be listed.                           *
19  *-----------------------------------------------------------*/
20 main(argc,argv)
21 int argc;
22 char *argv[];
23 {
24    char temp[120];
25    char device[30];
26    register int i;
27
28
29    pflinit();
30    argv++;                    /* bump past program name in args*/
31    if (argc > 1)
32       strcpy(temp,*argv);
33    else
34       strcpy(temp,".");       /* default directory */
35
36    if((dir = fopen(temp, "d")) == NULL) /* open the directory */
37       {
38          fprintf(stderr,"%s can't be read as a directory\n",temp);
39          exit(1);
40       }
41
42
43    strcpy(device,"@");  /* default device is data directory device */
44    if(temp[0] == '/')
45       {
46          i = 0;
47          do                      /
48             device[i] = temp[i];
49          while (isalnum(temp[++i]) || temp[i] == '.' || temp[i] == '_');
50          device[i++] = '@';
51          device[i] = '\0';
52          fprintf(stderr,"Device: %s\n",device);
53       }
54
55    /*-----------------------------------------*
56     *  Open the device containing the directory  *
57     *  we're about to list.                      *
58     *-----------------------------------------*/
59
60    if((disk = fopen(device,"r")) == NULL)
61       {
62          fprintf(stderr,"Error %d opening device %s\n",ferror(disk),device);
63          exit(1);
64       }
65
66    fread(&DirEntry, sizeof DirEntry, 1, dir); /* skip . entry */
67    fread(&DirEntry, sizeof DirEntry, 1, dir); /* skip .. entry */
68
69    /*-----------------------------------------*
70     * Read and format directory entries until EOF*
71     * Null entries are ignored by putEntry.     *
72     *-----------*-----------*-----------*-----------*/
73    while (fread(&DirEntry, sizeof DirEntry, 1, dir) != NULL)
74       putEntry(DirEntry.dir_name,DirEntry.dir_addr);
75
76    exit(0);
77 }
78
79
80 putEntry(Name,Address)
81    char *Name, *Address;
82    {
83       char CName[30];
84       long LSN;
85
86       if(Name[0] == '\0')
87          return; /* Null entry */
88
89       fixname(CName,Name); /* change OS-9 string (high-bit)
90                               to C format string */
91
92       ltol(&LSN,Address,1); /* make LSN useful */
93
94       printf("%s",CName);    /* reformatted file name */
95
96       expansion(LSN);        /* rest of the information */
97
98       return;
99    }
100
101 static struct fildes FD;
102
103 expansion(LSN) /* print everything interesting about a file */
104    long LSN;
105    {
106
107       if((seek(disk, LSN*256, 0) == EOF)
108          {
109             fprintf(stderr, "Disk seek error %d\n",ferror(disk));
110             exit(1);
111          }
112       if((read(&FD, sizeof FD, 1, disk) == NULL)
113          {
114             fprintf(stderr,"Disk read error %d\n",ferror(disk));
115             exit(1);
116          }
117
118       format_attr(FD.fd_att);
119       printf(" %u",FD.fd_own);
120       format_date(FD.fd_date,5);
121       printf(" %4d %1d",FD.fd_link, FD.fd_fsize);
122       format_date(FD.fd_dcr,3);
123       printf("\n");
124       return;
125    }
126
127
128
129 fixname(goodname,badname) /* convert from OS-9 string to C string */
130    char *goodname,*badname;
131    {
132       register int i;
133
134       i = 0;
135       do
136          {
137             *goodname++ = *badname & '\x7f';
138          }
139       while !((*badname++ > 0) && (++i <= 29));
140
141       *goodname = '\0';
142       return;
143    }
144
145 format_attr(attr) /* print file attributes */
146    char attr;
147    {
148
149       if(attr & S_IFDIR) /* is it a directory? */
150          printf("/ ( d");
151       else
152          printf("- (-");
153
154       if(attr & S_ISHARE)
155          printf(" ps");
156
157       if(attr & S_IOEXEC)
158          printf(" pe");
159
160       if(attr & S_IOWRITE)
161          printf(" pw");
162
163       if(attr & S_IOREAD)
164          printf(" pr");
```

```
165
166         if(attr & S_IEXEC)
167            printf(" e");
168
169         if(attr & S_IWRITE)
170            printf(" w");
171
172         if(attr & S_IREAD)
173            printf(" r");
174         printf("]");
175         return;
176      }
177
178   format_date(date,z) /* print a date in readable form */
179      char *date; /* yyyydd (hhmm) */
180      int z; /* number of entries in the date array */
181      {
182         char *month_name();
183
184         printf(" [%d Is 19102d",date[2], month_name(date[1]), date[0]);
185         if(z >= 5)
186            printf(" Id:102d", date[3], date[4]);
187         printf("]");
188         return;
189      }
190
191
192   char *month_name(n) /* return name of n-th month */
193      int n;
194      {
195         static char *name[] =
196            {
197               "Illegal month",
198               "January",
199               "February",
200               "March",
201               "April",
202               "May",
203               "June",
204               "July",
205               "August",
206               "September",
207               "October",
208               "November",
209               "December"
210            };
211
212         return(( n < 1 || n > 12) ? name[0] : name[n]);
213      }
```

```
1  #include <stdio.h>
2  #include <direct.h>
3  #include <modes.h>
4
5  static struct dirent DirEntry;
6  static int dir; /* path number */
7
8  /*  DirSqz */
9  /*      This program can be used to press the null
10         entries out of large directories that have
11         been hit with many deletions.
12 */
13 main()
14 {
15    long strt_ptr, end_ptr, backup();
16
17 printf();
18    if((dir = open(".", S_IFDIR+S_IREAD+S_IWRITE)) == NULL)
19       {
20          fprintf(stderr,"Error opening the directory %d\n",ferror(dir));
21          exit(1);
22       }
23    strt_ptr = sizeof DirEntry * 2; /* point past . and .. */
24    getstat(2, dir, &end_ptr); /* length of the file */
25    end_ptr -= sizeof DirEntry; /* point back from end */
26    end_ptr = backup(end_ptr,strt_ptr);
27
28    for(;strt_ptr < end_ptr; strt_ptr += sizeof DirEntry)
29       {
30          lseek(dir,strt_ptr,0);
31          read(dir, &DirEntry, sizeof DirEntry);
32          if(DirEntry.dir_name[0] == '\0')
33             {
34                end_ptr = backup(end_ptr, strt_ptr);
35                      /* leaved DirEntry with good data */
36                if(end_ptr <= strt_ptr)
37                   break;
38                lseek(dir, strt_ptr, 0);
39                write(dir, &DirEntry, sizeof DirEntry);
40                lseek(dir, end_ptr, 0);
41                write(dir, "", 1);
42                end_ptr = backup(end_ptr,strt_ptr);
43             }
44       }
45    exit(0);
46 }
47
48
49
50
51 long backup(end_ptr,strt_ptr)
52    long end_ptr, strt_ptr;
53    {
54       for(end_ptr >= strt_ptr;end_ptr -= sizeof DirEntry)
55          {
56             lseek(dir, end_ptr, 0);
57             read(dir, &DirEntry, sizeof DirEntry);
58             if(DirEntry.dir_name[0] != '\0')
59                break;
60          }
61       return(end_ptr);
62    }
```

"C" User Notes

Edgar M. (Bud) Pass, Ph.D.
1454 Latta Lane
Conyers, GA 30207

INTRODUCTION

This month's column provides additional followup information on problems associated with the 6809 Full C compilers, provides a few pointers on making C programs smaller and more efficient, and offers a few examples of the use of some of the string processing functions described in recent columns.

FOLLOW-UP

An earlier column stated that older versions of Introl C had size limitation problems with the linker. This has been corrected, at least as of v1.5.0. The problem with Introl C assuming that the FLEX system drive is always number zero has been corrected, as of v1.5.4. Occasional "branch out of range" problems remain in v1.5.0, but may or not remain in v1.5.4, as they have not yet reoccurred in my testing.

Most of the earlier problems reported for the McCosh series of C compilers have been corrected, as of v25.2.8. However, there are a few remaining, and they are covered briefly below.

Windrush C (for FLEX) does not honor backspace and line delete on line-oriented input (gets, fgets, scanf, fscanf, ...) from the terminal. This is a real nuisance, as the programmer must provide a routine to edit each input from the terminal before further use if any form of line correction is desired, making "scanf" essentially useless. The other (non-FLEX) versions of McCosh C may share this problem, but have not been tested. Indirection of the standard input with the FLEX I prefix-type command works, except that input defaults back to the terminal when the input file reaches end of file. The user must then enter a control-D to allow

the program to continue. Redirection of the standard input with "<" and redirection of the standard output with ">" are not supported in Windrush C.

The following program does not compile, but generates "Compiler Storage Error", at least in the Windrush version of McCosh for FLEX:

```
main()
{
    int c1;
    while ((getchar(&c1, 0), c1) != '\n');
}
```

McCosh C limits the expanded version of a C source line to about 127 characters. This is usually a nuisance problem both when writing new programs and when attempting to port them from other systems. It is not difficult to fix a statement which violates the line length limitation, unless it contains an expanded string longer than 127 characters, as most lines may readily be broken into more than one line. It is just a nuisance, especially when it occurs often, such as in a program which uses a large number of macro calls. The following program cannot be compiled with any of the McCosh C compilers because of their line length limitation:

```
#include "ctype.h"
main()
{
    if (isprint('a')||isprint('b')||isprint('c'));
}
```

A member of a structure is designated by a construct of the following form:

```
    structure-name . member-name
```

according to K & R. But K & R is silent on the situation in which member-name is not a member of structure-name, but is a member of another structure. Since this is almost always a coding error, most C compilers detect this situation. The McCosh C compilers will compile the following program, but the Introl C compilers will not:

```
#include "stdio.h"
struct x { char x1; } xx = { 'x' };
struct y { char y1; } yy = { 'y' };
main()
{
    printf("%c %c %c %c\n",
        xx.x1, xx.y1, yy.y1, yy.x1);
}
```

Although the allowance of crossed member-name references is harmless in this case, it can be disastrous in many cases, and most C programmers would prefer for the C compiler to flag crossed member names as syntax errors. If a given C compiler allows crossed member-name references, it should be documented as a warning to the C programmers using the compiler for development.

As noted in earlier columns, it is definitely not sufficient for a C compiler manual to state that it works "just like K & R". This is illustrated in cases such as the crossed structure names and members, in the order of evaluation of function arguments, in the implementation of the various data types (number of bits, signed/unsigned char, etc.), and in other areas not specified in K & R. Unfortunately, many C compiler manuals (not only those on the 6809) are silent on many of these important implementation details.

EFFICIENCY IN C PROGRAMS

The term "efficiency" has little meaning without reference to some basis of measurement. Most elementary programming texts discuss manners in which to make programs more efficient with respect to lines of code, run time, object code length, coding time, debugging time, etc.

C compilers have the generally well-deserved reputation for the production of machine code which would be judged fair to poor by experienced assembly language programmers. A few C compilers, such as Turbo C for the 8086, produce excellent machine code. The McCosh and Introl C compilers produce reasonably good machine code, even to the point of providing optimizers to attempt to clean up some of the most common and worst sequences.

Unfortunately, many implementations of C on microcomputers are based upon Ron Cain's original Small C compiler, which was intended to demonstrate that a subset of the C language could be implemented on a microcomputer, not to generate good code.

What can the C programmer do to attempt to make programs relatively smaller and faster, beyond the usual C code trimming and algorithm improvement? Often, very significant improvements in object code size may be realized by avoiding the use of the higher-level I/O functions such as printf, fprintf, sprintf, scanf, fscanf, and sscanf. Of course, avoidance of the use of these functions is sometimes difficult and time-consuming on the part of the programmer. However, the savings may be well worth the effort.

One problem with the higher-level I/O functions in many C compiler libraries is that they are so general in scope that they require the inclusion of most of the long, float, and double math libraries even if the program has no longs, floats, or doubles declared. For a reasonably short C program, these libraries may be several times longer than the object code length of the program itself. McCosh C compilers avoid some of these inefficiencies by requiring the user to request the inclusion of the long and float libraries explicitly for printf.

Unfortunately, printf is one of the most commonly used functions in the C library, so its exclusion in many programs may actually make them longer and more complex because of the C code required to replace each printf call. There is a simple solution which works in many cases. It is to code a version of printf which implements only the required subset of the capabilities of the general function, and avoids most of its overhead. Such a version of printf, fprintf, and sprintf appears below. It would require modification for use with many C compilers, as they pass arguments in reverse order, but provide an example of the direct inclusion of C library functions.

```
/*
**
**      Formatted print functions printf, fprintf,sprintf
**
**      They depend on the fact that some compilers push
```

```
**      function arguments in the order of occurrence.
**
**      Changes are required to use these functions with
**      c compilers which push arguments in reverse order.
**
**      These versions are non-standard since they require
**      the number of parameters as the last parameter.
**      However, this makes them much more portable.
**
*/


/*
**      Format and print to standard output
*/
printf (a, n)
int    *a, n;
{
    int    *fmt;
    char    buf[140];

    fmt = &a + n;
    _fmt (fmt, *fmt, buf);
    return (fputs (buf, stdout));
}

/*
**      Format and print to an i/o stream.
*/
fprintf (a, n)
int     *a, n;
{
    int         *fmt;
    char        buf[140];

    fmt = &a + n;
    _fmt (fmt, *fmt, buf);
    return (fputs (buf, fmt[1]));
}

/*
**      Format into memory at the address given.
*/
sprintf (a, n)
int     *a, n;
{
    int         *fmt;

    fmt = &a + n;
    _fmt (fmt, *fmt, fmt[1]);
}


/*
**      Internal function for printf, fprintf, sprintf
*/
_fmt (argptr, format, buf)
int    *argptr;                 /* point to arguments  */
char   *format,                 /* format string       */
       *buf;                    /* points to buffer
                                   to place results    */

char    c,                      /* temp character      */
        padchr,                 /* character to use for
                                   field padding       */
        *tstr,                  /* temporary pointer
                                   to a string         */
        tbuf[30],               /* temporary buffer    */
        ljust,                  /* flags to indicate
                                   left justification, */
        zpad,                   /* zero padding, and   */
        tsfull;                 /* temp string full    */
int     i,                      /* temp integer        */
        padlen,                 /* padding length      */
        len,                    /* length              */
        prec,                   /* field precision     */
        fldwidth;               /* field width         */

while (c = *format++)
{
    if (c == '%')
    {
        if (ljust = ((c = *format++) == '-'))
            c = *format++;
        if (zpad = (c == '0'))
        {
            padchr = '0';
            c = *format++;
        }
        else
            padchr = ' ';
        for (fldwidth = 0; isdigit (c); c = *format++)
            fldwidth = fldwidth * 10 + c - '0';
        if (c == '.')
        {
            prec = 0;
            while (isdigit (c = *format++))
                prec = prec * 10 + c - '0';
        }
        else
            prec = 10000;
        tstr = tbuf;
        tsfull = TRUE;
        switch (c)
        {
        case 'd':
            itoa (*--argptr, -10, tstr);
            break;
        case 'x':
            itoa (*--argptr, 16, tstr);
            break;
        case 'o':
            itoa (*--argptr, 8, tstr);
            break;
        case 'u':
            itoa (*--argptr, 10, tstr);
            break;
        case 'b':
            itoa (*--argptr, 2, tstr);
            break;
        case 'c':
            *buf++ = *--argptr;
```

```
                tsfull = FALSE;
                break;
            case 's':
                tstr = *--argptr;
                break;
            default:
                *buf++ = c;
                tsfull = FALSE;
                break;
            }
            if (tsfull)
            {
                if ((len = strlen (tstr)) > prec)
                    len = prec;
                if ((padlen = fldwidth - len) < 0)
                    padlen = 0;
                if (ljust)
                    buf = strncpy (buf, tstr, len);
                for (i = 1; i <= padlen; i++)
                    *buf++ = padchr;
                if (ljust == 0)
                    buf = strncpy (buf, tstr, len);
            }
        }
        else
            *buf++ = c;
    }
    *buf = NULL;
}
```

C PROBLEM

There are many ways in which to code a C program which translates upper case letters in a file to lower case and drops all control characters except carriage return. The one presented below will accomplish the desired result, using the translation functions described in the previous article in this column. Because of the continued strings used in this program, it cannot be compiled with the McCosh C compilers.

```
#include "stdio.h"
main()
{
    char s[255], *p;
    int n;
    /* the following statement clears the
        internal translation table to '\0',
        then maps the designated characters
        to their own character codes. */
    _str2map(2, "\n !\"#$%&'()*+,-./0123456789:\
;<=>?@[\_`abcdefghijklmnopqrstuvwxyz{|}~", "\0");
    /* the following statement changes the
        internal translation table to map
        upper case characters to lower case. */
    _str2map(1, "ABCDEFGHIJKLMNOPQRSTUVWXYZ",
                "abcdefghijklmnopqrstuvwxyz");
    /* this reads each line from a text file
        and checks for end of file. */
    while (fgets(s, 255, stdin) != NULL)
    {
        /* this translates the string just
            read, according to the internal
            translation table already built. */
        _strtrans(s, s, "", "", n = strlen(s));
        /* this outputs the translated
            string, skipping unwanted nulls. */
        for (p = s; n; --n, ++p)
            if (*p) putchar(*p);
    }
}
```

The next problem is to write a program which compresses multiple whitespace characters (space and tab) into single space characters in a text file. This may be done either directly or with the use of some of the string processing functions described in the last few columns.

EXAMPLE C PROGRAM

Following is this month's example C program; it sorts a file by a key. However, its primary point is not to illustrate a sort written in C, but to provide an extreme example of the necessity of the formatting of C programs.

Consider how much more readable and usable the program would have been if it were properly-formatted. Also, it will not compile directly with either Introl C nor McCosh C without several minor changes, since it was not written for either of them nor with portability in mind.

```
#include "stdio.h"
struct sort{struct sort*s_next;char s_buf[256];
};main(argc,argv)int argc;char**argv;{struct
sort*s,*s2,*bs,*bs2,*as,*as2;struct sort
*last_sort=NULL,*first_sort;int count_sorts=0,
n_sort,i,j,wrong;int rand(),srand();while(argc>
1){++argv,--argc;if(**argv!='-'){fprintf(stderr,
"sort: %s: no file argumentsn",*argv);}else
fprintf(stderr,"sort: %s: no options eithern",
*argv);}do{s=(struct sort*)calloc(1,sizeof(
struct sort));if(last_sort)last_sort->s_next=s;
else first_sort=s;last_sort=s;++count_sorts;}
while(gets(s->s_buf));--count_sorts;srand(time(0
));do{for(i=0;i<count_sorts;++i){for(s=first_sort
,j=0;j<i;s=s->s_next,++j);/*ideally we would loop
on rand() until the value fell into the range 0-(
count_sorts-1), but we will cheat*/n_sort=rand()
%count_sorts;for(s2=first_sort,j=0;j<n_sort;s2=
s2->s_next,++j);for(bs=first_sort;bs->s_next&&
bs->s_next!=s;bs=bs->s_next);if(bs->s_next==NULL
)bs=NULL;for(bs2=first_sort;bs2->s_next&&
bs2->s_next!=s2;bs2=bs2->s_next);if(bs2->s_next
==NULL)bs2=NULL;for(as=first_sort;as->s_next&&
as!=s->s_next;as=as->s_next);for(as2=first_sort;
as2->s_next&&as2!=s2->s_next;as2=as2->s_next);if
(s->s_next==s2){if(bs)bs->s_next=s2;else
first_sort=s2;s2->s_next=s;s->s_next=as2;}else
if(s2->s_next==s){if(bs2)bs2->s_next=s;else
first_sort=s;s->s_next=s2;s2->s_next=as;}else{
if(bs)bs->s_next=s2;else first_sort=s2;if(bs2)
bs2->s_next=s;else first_sort=s;s->s_next=as2;
```

```
s2->s_next=as;}}wrong=0;for(s=first_sort;
s->s_next&&s->s_next->s_next;s=s->s_next){if(
strcmp(s->s_buf,s->s_next->s_buf)>0){wrong=1;}
while(wrong);for(s=first_sort;s->s_next;s=
s->s_next)puts(s->s_buf);exit(-1);}
```

68000 USER NOTES

Phillip Lucido
2320 Saratoga Drive
Sharpville, PA 16150

I quit! Not writing the column, of course. I'm learning a lot, and having fun doing so. But from now on, I do solemnly promise to hold back on those previews of what to expect in the following column. Something else always seems to come up, and my scheduled plans are constantly shot to pieces.

Anyway, this month I will not be reviewing version 1.0 of OS-9. This column is going out a little early, as I get ready to go to Microware's OS-9 seminar, so the new release isn't here yet. Further, my impassioned call for program standards will probably have to wait for some indeterminate time in the future (I'd say next month, but my solemn promises generally hold for a minimum of two paragraphs).

Egg on My Face Dept.

What will I use this month's column for, then? Retractions and corrections, of course! Nothing serious, but a few points from last month turned out to be incorrect.

Last month, Kirk Anderson had a question concerning the use, or lack of use, of the '#' memory allocation qualifier. Under OS-9/68K, specifying a large buffer for a utility like copy is done with the command 'copy -b=20K ...' instead of 'copy #20K ...', as you would do with OS-9/6809. I assumed that the buffer is allocated with an sbrk() call, which contiguously expands a program's data allocation with an F$Mem OS-9 service request, and had some reservations about such a practice, especially in a Level 1 system.

While talking to people at Microware, I was told that an sbrk() call is not used. Instead, there is a new C system call, ebrk(), which is used to request more data memory. Unlike sbrk(), the ebrk() call does not attempt to allocate memory contiguous with existing data memory. Instead, it uses a call by the name of F$SRqMem (System Request Memory), which allocates memory without regard to it's location. The F$SRqMem call is not available to user programs under OS-9/6809, where it is a reserved system service request, but under OS-9/68K it is a user request.

This difference between the 6809 and 68000 versions of OS-9 is not particularly important to most programs. It does mean, though, that programs which build large tables in memory, like compilers or assemblers, will not have the problems under Level 1 OS-9/68K that they would under Level 1 OS-9/6809.

Last month, I also mentioned a sorting program that I was writing. The program should have taken most of it's time reading and writing the disk, so the 6809 and 68008 versions should have run equally fast. Instead, the 6809 version took 27 minutes, with the 68008 version taking 10 and a half.

Well, obviously the 68008 can't be that much better. Instead, the times reflect another difference in memory allocation between OS-9/6809 and OS-9/68K. My program uses the buffered I/O routines in the C library, which use buffers to quickly save file input and output, then transmit the results in 256 byte chunks to OS-9. This runs much faster than sending data character by character through OS-9, with I$Read and I$Write calls, since there is significant overhead with each actual call to the OS-9 kernel.

Under OS-9/6809, these buffers are allocated using ibrk() calls, which use the memory in the original data area. If enough memory is not immediately available, then the associated file is set unbuffered, so the OS-9 kernel is called for each character. This is what happened with my sort program, which has about 8 files open at once, and needs 2K of memory just for the I/O buffers. The C compiler includes a command line option, -m, to increase the initial memory allocation, but if it is not used, then there is only enough memory for at most 3 files. Thus, most of the files in the program ended up being unbuffered. The long running time for the 6809 version was entirely due to the constant overhead calling OS-9.

Under OS-9/68K, the I/O buffers are allocated using ebrk() calls, so there is no problem with not setting the -m option when compiling. After properly compiling the 6809 version, with an option of -m=10 to allocate space for up to 10 additional buffers, the program ran in about the same time on both processors.

Why the 68000?

There is something I probably should have talked about some time ago, to wit, what is there to recommend the 68000? There are actually two questions here. First, in what ways are the newer 16 bit microprocessors better (or worse) than the 8 bit machines, and second, how does the 68000 compare with other 16 bit processors?

16 bit processors offer two main advantages over the 8 bit processors: an increased address space and a wider data bus. The address space of a processor is the amount of memory which can be directly addressed, without first going into any software bank switching or the like. This is determined by the number of address bits on the chip. 8 bit processors generally have 16 address bits, which allow them to directly address 64K bytes of memory. 16 bit processors vary, but most have at least 20 address bits, for a 1 megabyte (1M) address space, like the 68008, or 24 bits like the 68000, for 16M of directly addressable memory.

For most of the time since microcomputers made their appearance, 64K of memory has been enough. Increasingly, though, programs have been appearing which either require, or run much better, in a large amount of RAM. It is possible to run such programs in an 8 bit machine, by using such memory techniques as page mapping, like that used in OS-9/6809, or by writing a program to run in overlays and keep data on the disk. However, this complicates the program, and distracts the programmer from the program's true function. Often, the program simply doesn't get written in its most powerful form, if at all. By removing this 64K limitation, 16 bit micros make it simpler to write these large programs.

In addition to a larger address space, 16 bit chips tend to be more efficient at processing data. They do this by being able to perform their various machine language operations on larger bit-groupings of data. On an 8 bit micro, arithmetic is generally performed on byte, or 8 bit, data, while 2 bytes, or 16 bits, is the basic data size for the larger processors. Being able to handle more data per instruction helps a program to run faster, since fewer multi-byte operations need to be performed.

Now it is fairly obvious that 8 bit micros won't disappear overnight. For most jobs they are more than powerful enough. Furthermore, 16 bit microcomputers are likely to remain more expensive than 8 bit designs for some time, with their larger memory requirements and newer chip sets. Because of this, for instance, the 68000 is unlikely to eclipse the 6809 to the same extent as the 6809 has done so to the 6800. What is likely to happen, though, is that new programs will be developed for the 16 bit chips which will simply be to difficult or large to transport back to the older computers. If you want the speed and can handle the added expense, then a 16 bit microcomputer may be worth it.

Which particular 16 bit (or 32 bit) chip is likely to dominate the market? Unfortunately, the decision will depend more on which chip IBM or AT&T choose to put in their machines than on which chip is more powerful. Still, I can always *hope* for the best, and explain why I prefer the 68000 (surprise, surprise).

My exposure to 16 bit micros has been mostly limited to the 8086 and the 68000. Thus, I can't really say much about the Z8000 or any of the others, but these seem to be minor players in the game, anyway. The 8086 is obviously the front runner, with its use in all of the IBM PCs and PC clones, but this is due more to its earlier appearance on the scene. The 68000 clearly has a more powerful design.

First, the 68000 might better be described as a 32 bit micro. There are two measures of bit size in a processor. The normal one is the width of the external data bus, which is the number of bits which can be written or read at a time. This is 16 bits for a 68000 or 8086, 8 bits for a 68008, 8088, or 6809. The other bit size, which may be more important, is the width of the internal data bus, reflected in the size of the general registers used for most arithmetic in the machine. For the 68000 and the 68008, this is 32 bits, while it is only 16 bits for the 8086 and 8088. As I said above, the ability to handle larger chunks of data, which depends on the internal data bus width, is a strong factor in the speed and performance of a processor.

Second, the 8086, in attempting to stay upwardly compatible with the 8080, uses a segmented addressing scheme for addressing over 64K of memory. In the 8086, addressing memory requires two different values. One value, held in what is known as a segment register, points to a base address which is anywhere in a 1 megabyte range. The segment register is 16 bits long, and points to the address formed by appending 4 bits of zero to the end of the value in the register, creating a 20 bit memory pointer. The second value used in addressing is the offset, which might come from an index register or be part of the instruction (like extended addressing in the 6809). To perform the actual memory access, the shifted segment register value is added to offset, giving a final 20 bit address.

This method has some advantages. For instance, the segment register is generally loaded only once per program or once per subroutine, so addresses from that point on are only 16 bits long, the size of the offset, reducing the size of the object code. The disadvantages are quite serious, though. If you look closely, the segmented addressing is just another version of bank switching, albeit somewhat more manageable than in the 8 bit/64K situation. As a result, it is impossible to address more than 64K at a time without manipulating the segment registers. There are separate segment registers for data, stack, and program access, but there is still that 64K problem within a single segment.

How is the 68000 different? Motorola chose the simplest method of memory addressing, by making all addresses 32 bits long. There is no memory segmentation required, since a single instruction can directly access any byte in a 4 gigabyte (that's 4 billion bytes!) range. Obviously, programs for the 68000 will often be larger than equivalent programs for the 8086, since larger addresses (and larger instructions in general, by the way) have to be kept in a program. But what must be remembered is the fact that, from now on, memory is cheap, especially when compared to programmer time. A program might now be 40K long instead of 32K, but if there is 256K of memory in a computer, so what?

The 68000 is not entirely without blemishes. While 32 bit addresses are used, constant offset indexing is still limited to a 16 bit range, which is not significantly different from the segmented addressing of the 8086. This is less troublesome here, since 64K is generally sufficient for named variables, which are the types which will be addressed with constant indexes from a base register. Large data tables using pointers to link table elements are not affected, and can be as large as required, up to the bounds of available memory. Also, I have heard that the next processor in the 68xxx family, the true 32 bit 68020, will allow full 32 bit offsets for indexing.

Vacation!

That's enough for now. I'm off to the OS-9 seminar now, with a nice lazy (and well deserved) vacation to follow that. Next month I should (not will - should) be answering some mail that has come in, as well as telling you what transpired at the seminar. Bye.

- - -

SINGLE BOARD COMPUTERS-6809

SINGLE BOARD COMPUTERS - 6809

Recently we received for review 3 different single board 6809 computers. All three are 64K systems, with 56K standard per FLEX™ convention. All three boards run FLEX, two have also licensed OS-9™ level one. The two FLEX systems recommend that you purchase FLEX from your favorite source and use their modified drivers. Essentially this requires most any FLEX.COR and append the drivers to make a bootable FLEX system. Some consideration should be given to certain SWTPC FLEX versions, however, all can be made to work. Specifics will be covered in the review of each system.

The three systems we will look at are:

1. The PT-69™
   Peripheral Technology
   3760 Lower Roswell Rd.
   Marietta, GA 30067
   404/973-0042

2. ST-2900 System™
   Sardis Technologies
   2261 E. 11th Ave.
   Vancouver, B.C.,Canada V5N 1Z7

3. The 6809 "Uniboard"™
   Digital Research Computers (of Texas)
   P.O. Box 461565
   Garland, TX 75046
   214/271-3538

Notice should be taken that we will review each system in the order of A-Z. Why? Well they all have certain strengths and weaknesses, as we see it. Also we ended up having no particular favorite, as each has certain merits not available to the other two. All three are advertised in 68 Micro Journal and are running either in our offices or our lab (meaning they have been tested and accepted by our standards). All three perform well. Any one of the three when combined with disks and a CRT or keyboard and monitor (depending on the system) make an excellent, general purpose or specialized 6809 64K computer. The boards alone make great and very economical 6809 controllers or stand-alone systems. I see an unsurge in 6809 activity due to the economy and availability of these systems running all the popular 6809 disk systems and software!

THE PT-69

The PT-69 is completely mounted on a single glass epoxy board, 6 1/2 X 5 1/2 inches in size. The board is solder-masked and double sided plated through. Also the system can be obtained complete with or without cabinet, power supply and 5 inch disk drives, 40 or 80 track. However, we will only review the board without power supply or disk drives. We added our own drives and mounted the board in a Heath H-19 CRT terminal that we had in our lab. Also the Sardis system is mounted in a Heath H-19 CRT terminal. Both these systems have half size Qume 5 inch DO OS disk drives 40 track, installed in the CRT terminal also. Everything in one box.

I have long seen the need and attempted to get some of our present 6809 computer manufacturers to make a similar system. A very accurate survey some two years ago indicated that many of you wanted such a system. Only SWTPC and WaveMate have done so.

WaveMate blew it by making the hardware and software dependent on a double density disk directory for FLEX. All normal FLEX systems use single density directories, for both single or double density format. Had they listened I sincerely believe that they would have had a winner, but now *only* SWTPC advertises a system in a desktop configuration (X-12+), and I understand it is doing quite well. However, by utilizing a CRT terminal similar to the Heath H-19, which has provisions for disks also, the entire system can be in one package. And that is the wave of the future, something we should have done years ago. Now with two of these, desktop complete systems are possible. With the other the size of the board is slightly too large, due to features not available on the other two. Remember, I said advantages and disadvantages.

Now, as to the Heath H-19, it is no longer in production, but many are advertised as usad and at very good prices, so it should not be too difficult finding a low price used one or a similar type. Should any of you out there have a used Heath H-19 for sale, please let me know as I am certain I will be receiving many inquiries for availability of used ones.

On to the PT-69. Basic overview:

56K RAM useable
4K EPROM - 4K I/O
2 8 bit parallel ports (6850)
2 RS232 serial ports (6821)
1 Mhz 6809E processor
Double density, double sided 40/80 5" disk drives
Available with optional CRT, cabinet, power supplies and drives
Time of day and calendar clock (146818)

Although a kit is available, we received the review system built and tested. It required about 4 leisure hours to install the system in the CRT terminal, wire the serial port and parallel port to the DB25 connectors on the back of the CRT terminal, wire the communications port to the terminal, drill a couple of holes on the support inner frame of the terminal to mount the board, rob power from the terminal and go. Also a small power supply for the disk drives should be built or purchased and installed.

Because our review system came prebuilt and tested it came up online without a hitch, first time. We have a second one that we use for maintaining a mailing list and it has functioned flawlessly for over six months, 8 hours a day.

Operating Systems

The system runs both FLEX and OS-9 level one. And this brings up an interesting point. The disk controllers sold by Peripheral Technology function as SWTPC DC2-4 disk controllers. Therefore, the OS-9 configuration for the PT-69 system should also run on any SWTPC system using SWTPC DC2-4 5" disk controllers. Now for the many of you who have expressed a desire to run OS-9 on your SWTPC I would suggest you contact Peripheral Technology for the particulars.

The FLEX version uses a monitor that has entry points that are the same as the SBug-E monitor from SWTPC, less any 'DAT' functions (remember it is only a 64K system and needs no DAT). I understand an expansion model will be available in the near future and will allow for extended addressing. Then a DAT type monitor such as the SWTPC will run in the system. The documentation indicates that FLEX version 9.1 from TSC will run as well as most SWTPC FLEX versions. Almost any FLEX.COR should work, except some SWTPC versions that have relocated some parts of the .COR. Also available is disk drivers to make the TSC version run 00 DS.

The Monitor PT-MON

The monitor has entry points that coincide with those defined by SWTPC when the first 6809 CPUs became available. By sticking to these standard entry points most all software runs unaltered. For many this is not only convenient but a MUST. Patching software that talks directly with I/O devices or monitor calls that are either different devices or different entry points can make for some long debugging sessions! We have not had to alter or change one piece of software running on our day-to-day PT-69 office system.

Monitor Commands:

Alter Accumulator-A
Alter Accumulator-B
Alter Conditional Code Register
Alter DP
Alter U Stack Pointer
Alte X Register
Alter Y Register
Set Breakpoint
Dump Memory (both hex and ASCII)
Execute Program
Find Data (two hex bytes)
Continue Execution of Program
Jump to Subroutine
Initialize Memory (any char 0-F)
Load Tape (S1 format)
Memory Examine and Change
Punch Tape (S1 format)
Test Memory
Register Dump
Boot Floppy Disk
Remove Breakpoint

The Monitor occupies memory from $f800 $ffff. The stack pointer is placed at $c0ff except when using a version of SWTPC FLEX 2.8:3 or higher, else it is at $dfc0. Monitor routines are entered by indirect jump calls. The more significant ones are:

F800 - Monitor - Re-enter monitor
F802 - Nextcmd - Re-enter monitor and prompt
F804 - Inch - Get input char from terminal no echo
F806 - Inche - Get input char from terminal with echo of char
F808 - Incheck - Check for input char
F80A - Outch - Output char to terminal
F80C - Pdata - Print data string
F80E - Pcrlf - Print carriage return and linefeed
F810 - Pstrng - Call Pcrlf then Pdata
F812 - RTS - Null, included for S-Bug-E compatibility
F814 - Out2hs - Print two hex char
F816 - Out4hs - Print four hex char
F818 - In2hex - Input two hex char
F81A - In4hex - Input four hex char
F81C - Out4hex - Print four hex char in 'X' reg
DFC2 - SWI3
DFC4 - SWI2
DFC6 - FIRQ
DFC8 - IRQ
DFCA - SWI

Memory Map

0000 - 0FFF RAM
E001 - E002 ACIA
E004 - E005 ACIA
E010 - E013 PIA
E014 - Drive select register
E018 - E01B WD2797 (disk controller)
E050 - E05E RTC (clock)
F800 - FFFF Monitor (2716)
F000 - FFFF Monitor (2732)

Conclusion

No provision are made for 8 inch disk drives, however, the disk controller provides for 8" drives as well. A good hardware type should experience little trouble in adding 8" capability. How about an article someone who has or will do it! But, with 80 track drives, why?

Eleven plugs and jumpers are provided to select baud rates for both ACIAs, terminal, printer/modem, floppy drive cable, PIA, power, printer CTS enable, 2716/2732 select, reset, disk controller test input and real time clock battery connector. We use NI-Cads and the date and time is always there at power-on.

Baud rates hard wired are 300, 1200, 9600 and 19200 for both ACIAs. Other rates can be jumper wired in.

The documentation is complete with schematic drawings, parts placement, parts list, cable connection charts and complete manufacturers spec sheets and booklets on each of the major components, 6809, 6821, 6850, 6883, 146818 and WD279X disk controller.

The nice part about this system is that you can run practically all FLEX and OS-9 level one software without modification. It is simple to install and get running. The I/O devices are those normally looked for by most software and are at normal memory addresses. We could find nothing to complain about except that the documentation is certainly not 'Heath' quality. But then neither is the other two, so I guess that runs even. It is sufficient to get the job done, but it will be simpler for those who have some experience with building kits or boards and wiring cables and harnesses. Anyone should be able to do it by detail study of the total package of documentation.

The price of the board - wired and tested is:

$299.95 less power supply and cabinet
(and of course disk drives)

The system with power supply and cabinet is:

$399.95

The complete system with power supp y and cabinet, two 5"disk drives DD,DS 40 track is:

$999.95

With the cost of the PT-69 board, a used CRT terminal and two 5" disk drives DD DS, the total system cost should be $1500 or less, and that is the advantage of the new wave of single board 6809 computers, cost, compactness and semi-portability (with the Heath terminal the total weight is about 45 pounds).

For those applications demanding additional I/O, hard-disk and other peripheral interfacing then one of the larger S50 Bus system will be required, but for many this is the way to go. And it is good for the industry, for experience has shown that satisfied small system owners eventually graduate to larger and more complex 6809 systems rather than go off to the 'other side' (who wants to learn new languages, buy new software and essentially start all over?)

See Peripheral **Technology** Advertising for additional specs and ordering information.

Next month a review of the ST-2900 System 6809 single board computer from Sardis Technologies.

- - -

COBOL

COBOL

About the only language that I had never gotten to work with (until last week) was Cobol. About a week ago, "Crunch Cobol" from Compusense arrived for me to look at and perhaps review. Well, after a week I have some initial impressions of both the language and the particular implementation, but a week is certainly not enough time for anyone to have become proficient in any language, so I really don't feel qualified to do a thorough review of the Compusense package. However, I suspect that Cobol might be new to most of you readers as well as to me, so some first impressions of the language interspersed with some comments on this particular implementation might be of some interest.

As do most of the compiler packages, this one arrived with a manual that clearly indicates that it does not contain a complete tutorial on the language. I went to the local bookstore and found a larger than expected selection of books on Cobol. One of them was wrapped in plastic (I think to keep some additional sheets provided with it from becoming lost) so I couldn't look at it. Now I'm not ready to spend $20 on a book if I can't at least leaf through it and see what it contains. I settled for a book called Structured Cobol, A Self Teaching Guide, by Ruth Ashley. Nearby was another book by the same author that seemed a little older.

To quote from the introduction, "Structured COBOL deals with the COBOL language -- the same COBOL that programmers and computers have been using for years. 'Structured' here refers to programming, and, as such, is independent of the COBOL language. Structure is an approach to programming in which we are concerned with clarity as well as effectiveness."

My very first impression was that COBOL's author(s) searched hard to find keywords that are as long as possible for each function. Print and Write are commonly used in other languages as keywords to cause output of what follows. These are both 5 letter words, too short for COBOL, which uses DISPLAY for the same purpose. An example program had the line: "DISPLAY "*** END OF FILE ***" UPON CONSOLE. The UPON CONSOLE qualifier keeps information such as this on the CRT terminal even if the other outputs are redirected, as to a printer. In an effort to keep the programmer thinking in terms of writing programs in "plain English", COBOL doesn't have "procedures" but rather uses the term "paragraphs". Statements are called "sentences" and they end with a period (of course).

Unfortunately, the book is too much "self teaching" oriented, following the question and answer approach all the way through. The example programs are the most useful information contained. Though the author made an attempt to organize the information roughly by subject it is very difficult to find some of the information. A paragraph, being analogous to a procedure in Pascal or PL/ or a function in "C", ought to be a rathe important topic, right? The word Paragraph doesn't appear in the alphabetical index at the end of the book, and I had some trouble with the syntax of my first attempted paragraph, which turned out to be more complex than any in the examples in the book.

I've recently decided that I would compare the file handling in several languages, and so I wrote a short program that reads a text file, converts all upper case characters in it to lower, and writes the result back to another file. I thought I would try it in COBOL. First attempt caused 31 errors to be reported, but I'm getting ahead of the story... More on that later.

I found, after a few evenings of reading the book and the instruction manual that came with the Crunch Cobol, that I had some definite impressions of COBOL as a language, but not yet a very good idea of how good the Compusense implementation is. First the impressions, then a discussion of the implementation. I found that COBOL is VERY good in the area of defining a RECORD. Each field is described character by character, indicating which positions contain alphabetic characters and which contain numeric. (I'm speaking initially of how a record in a FILE is described in the header sections of the program). The formatting capabilities for the generation of output strings for reports is considerably more comprehensive. These capabilities must be what inspired the extensions of BASIC in the area of the "PRINT USING" facilities. You can specify separators for numbers, such as commas or slashes (for date information). You can have leading zeros suppressed or present. You can fill the leading zero columns with "*", or blanks. You can have a dollar sign in a fixed column, or floating to put itself before the first non zero digit in the result.

Formatting is done by means of a PICture. For example a record containing name and phone number information might look like this

```
01  CUSTOMER-RECORD
    02 C-FIRST-NAME          PIC X(12).
    02 C-LAST-NAME           PIC X(15).
    02 C-PHONE-NU            PIC 9(10).:
```

In the "PIC" area, X indicates alphanumeric (character) information, and 9 indicates numeric. In the present case, the phone number could just as well be defined as a field of characters. In the output formatting section of the program, several other symbols are used to insert commas, place the decimal point, etc. In the present program, I wanted to input a line of text and modify it.

I found the Compusense implementation to be rather complete. The manual describing the use of the compiler is clearly written, and I had no problem running it. To my horror, my first attempt at the program resulted in a total error count of 31. Since the whole program was about 40 lines long, it looked pretty bleak. I soon found out that the source program format may not be quite as "free form" as with some of the other languages, and that I hadn't indented the statements enough from the Label and Heading column. Indenting one column further brought the error count down to ten. Another pass straightening out some syntax (missing period at end of some statements and headers) brought the error down to just one. It seems that no arithmetic is permitted on a character. You may only perform arithmetic on numeric fields.

I was temporarily set back trying to figure out how to convert "A" to "a" without simply adding or ORing in $20 to the value. The test IF CHAR NOT < "A" AND CHAR NOT > "Z" worked fine to sort out the characters to be modified, but the ADD $20 TO CHAR resulted in an error. (Note that comparisons in Cobol may use only one symbol and optionally the word NOT. >= translates to NOT <, and <= translates to NOT >.) Adding decimal 32 didn't work either. There seem to be no "conversion" functions to convert a character to a number so arithmetic can be done on it, either. The only other possibility I could imagine was a section of code containing 26 IF statements:

```
IF CHAR = "A" MOVE "a" TO CHAR.
```

Repeat this all the way through "Z", and you have something that should work. Wrong again. I found that when I tried to set up the IN-FILE record to be one character and read one character at a time to process it, I only got the first character of each line. Apparently COBOL is like BASIC in that respect. It reads to the first carriage return and puts what will fit, into the defined record, throwing everything else away. What to do now?

```
01  IN-LINE
    02 IN-CHAR          PIC 9(80).
```

I declared the record to be 80 characters. In the working data section I declared a structure to which to

move the line read from the input file:

```
01 W-LINE
   02 CHAR OCCURS 80 TIMES.
```

That peculiar syntax sets up an array of characters of dimension 80. Now I could READ a record from INPUT-FILE, and MOVE IN-LINE TO W-LINE. Then I could index through W-LINE as an array of characters, modify the characters and return them to W-LINE, then move W-LINE to OUT-LINE and write it to the new file. Simple? Yes but there is still a problem. COBOL is obstinate about having records of fixed length, even in a sequential file. I had to fill IN-LINE with blanks by moving a blank line into it, read the record, move it to W-LINE, modify it, move it to OUT-LINE and write it to the output file. A quick LIST of the output file after running the program, which by this time had no compile e rors, indicated that it worked fine. I then discovered that the output file was three or four times as big as the input file. A quick dump of the disk file showed that each output record was 80 characters long, and that after the text ran out, the record was padded with nulls ($00) with a CR ($0D) at the end of the record to serve as a separator.

Further, the program, because of the average of 13 IF comparisons for each character, took about three minutes to run on the source listing of itself. I split up the decisions with some tests to get the comparisons at least to the proper quarter of the alphabet with a couple of preliminary IF's (see the listing here), and the whole thing stopped working. It turned out that adding labels within my SUBSTITUTION paragraph, fooled the compiler into putting the return just before the first embedded label. I had to declare "SUBSTITUTION SECTION." In ·der for it to accept the several labels and consider the paragraph as one. The Compusense manual indicates that I probably should have been able to use "PARAGRAPH-SUBSTITUTION." as a heade to identify the paragraph, but that didn't work, though no compiler error was flagged. As I indicated earlier, the book doesn't even contain a reference to the word paragraph in its index. None of the example programs in either the book or the manual included multiple labels, so I am in the dark as to why what would seem like the more logical identifier didn't work at all.

It seems that the fixed record length even for sequential files, made the program work very inefficiently. I tried cutting off the character match process by stopping at the CR in the input line, but it appears that the CR is not included in the input line, nor transfered to W-LINE. I also found that if I didn't "blank out" IN-LINE each time, the output line written to the output file contains the tail end of the previous lines longer than the current one. I did find that running the output file through any of the editors, stripped it of all the extra nulls and the output file from the edit, was the same length as the input file.

At any rate, with the 26 IF statements broken down into groups of 6 and 7, the execution time went from three minutes to just under two minutes.

At this point, I'm willing to say that COBOL is not intended for use in "character manipulation" applications. It did the job, though rather inefficiently. The fact that all reco ds must be the same length for sequential files, removes the usual compactness advantage of using sequential files, and any data file might as well be the random access type, since there is no accompanying less efficient use of disk space to offset the quicker access and the capability of adding to or changing records in a random access file. This limitation, also makes COBOL unsuitable for use in manipulating text files, as for text editing applications.

As a point for comparison, I ran the Whimsical compiler version of the "LOWER" program on the COBOL source file, and it ran in ten seconds. I was impressed with the small amount of code generated by the compiler. However, I should point out that this is a "P-code" implementation that runs with a "runtime package".

Obviously, since I am no expert on COBOL, I am not in a position to make any absolute judgements on this package. I will say that it appears to be a fairly complete implementation of ANS COBOL. The manual doesn't indicate the precision of the arithmetic package, which I obviously didn't check with the program here. I found the error messages to be of little help. When the error line was an ADD statement, the error message was "Syntax e r · in ADD statement". Actually the error message is a code, and you must look up the code in the manual. The next error was in a MOVE statement, and the error message was "Syntax error in MOVE statement". A simple "SYNTAX ERROR" would have done, since I could see what type of statement the error line contained. The error line is output with a carat pointing at the approximate location of the error in the line.

As with most modern compilers, my few errors produced a large number of error messages and the correction of any one error significantly reduced the count, so that I quicky had an error free compile. Compile time for the final program on a 2 MHz system (all times given here are for that system) was around two minutes. The compiler has the usual set of options, not unlike the TSC Assembler that most of us have. You can create an output file or not, create a listing to the terminal or not, list to the printer, etc. There are some facilities for including TRACE information in the compiled output, and for including optional debug statements in the output code.

One nice feature of the compiler is an extension that allows the program to "parse" the FLEX command line for filenames that follow the command that invokes the program. Up to 5 filenames may be included on the command line and associated with the logical filenames used within the program. The syntax: SELECT INPUT-FILE ASSIGN TO FILE-1, associates the logical filename INPUT-FILE with the first file on the command line, etc. Alternately, a literal filename may be included as in SELECT INPUT-FILE ASSIGN TO "WTESTFILETXT", and it will cause TESTFILE.TXT from the Working drive to be associated with INPUT-FILE. The name of a variable that contains a string that is the name of a file may also be used.

I think, after seeing the capabilities of this language in the area of defining records, and formatting outputs, I would use it to write specific data handling software. I can see that a lot of the thinking that went into the currently available database management software came from COBOL. Since the language was designed to handle data processing needs, it really ought to shine in such applications.

It is obviously seriously lacking in the capabilities that would make it a good language in which to write systems software or major number crunching programs. One notable lack in COBOL is the availability of "local variables", and right along with that, the capability of passing parameters to "paragraphs". All variables are GLOBAL, and the various paragraphs do their thing by modifying the global variables. In spite of this limitation, the language has enough statement types to allow pret well structured programs.

Though I have used GO TO in the example program, it would not be hard to eliminate it in most instances. Some of the so-called GOTO-less languages ("C" for example) have eliminated the GOTO by calling it something else (sometimes the keyword BREAK is used). Though I realize that BREAK is limited to causing exit to the statement after the loop in which it is used, it really is a GOTO. I've used GO TO just as I would use BREAK, in the SUBSTITUTION paragraph, allowing me to skip the remainder of the code after I have performed the necessa y action.

Well, though this turned out to be a discussion and "sort of a review", it is getting very long, and I am going to have to quit here. If there are any COBOL programmers out there, I suppose one or more of them will see that I have missed an easy way to convert upper to lower case, please let me know how dumb I am. I'll print any comments you might have. If anyone out there knows of a GOOD book on Cobol, please let me know about it also.

Reviewed by: Ron Anderson

Compusense Crunch Cobol:

Available from: SOUTH EAST MEDIA
Call Toll-Free - see advertising this issue

Price: Regular $199.00

Special Introduction - $99.95

- - -

PRODUCT REVIEW DATASYSTEMS 68 COMPUTER

PRODUCT REVIEW DATASYSTEMS 68 COMPUTER
** NOW DIGITAL RESEARCH COMPUTERS (of Texas) **

Stewart D. Lyon
19943 Arminta St.
Winnetka, CA 91306

INTRODUCTION

Like many other hobbyists, my first experience with computers was with the Motorola D2 kit. I spent a lot of time hand coding that thing. At one time I could write programs without looking at a manual--I had the entire 6800 code memorized! I gradually built up the D2 kit to include a video monitor, 32K ram and a real keyboard--no more hex keypad. The next big step was a TRS-80C. Finally, I acquired a disk system and FLEX. Actually the TRS-80C was my son's Christmas present. To settle a lot of family squabbles, and because I was tired of the TRS-80C screen and keyboard, I started looking for a "real" 6809 computer. That's when I ran across DATA SYSTEMS' ad in this journal. By this time, I had parlayed my 6800 experience into building several dedicated computers for rocket payload control; consequently, I felt this was a project I could handle. My twenty or so years as a ham and as an engineer added to my confidence. So, off went my order.

The boards I ordered from DATA SYSTEMS 68 were:

a) CPU-6809 (the processor board)
b) 6845 Video Display
c) DRAM-64K Ram board
d) FDC-50 Floppy Disk Controller
e) MULTI-I/O Board
f) DUAL SERIAL INTERFACE Card, and
g) THE MOTHER BOARD

What arrived was a beautiful set of boards and about 3/4" of manuals. The boards are glass epoxy with double sided copper and solder masking. Most of the boards are 1/16" thick except for the mother board which is 3/32" thick. A separate manual is provided for each board, each including a short section on theory, a brief construction guide (this ain't no Heathkit), a schematic, parts list, assembly drawing and, where necessary, some software.

DS68 assumes that you have access to another FLEX system with an EPROM burner and a suitable monitor such as SWTP's SBUG E. Also you'll need a General Version of FLEX. Additionally, you will require a power supply, keyboard, video monitor, enclosure, disk drives and probably a printer. All DS68 is selling you is blank boards.

Assembling the boards was easy (once I gathered up all the parts), getting to work was fun, and building the enclosure was a pain in the neck. I really have a problem punching oblong connector holes. After about six months of effort, I have a computer that is as contemporary as any and didn't cost a fortune. It was a lot of work but that's what a hobby is all about.

Following is a board by board description of the DS68 computer and my impressions.

CPU BOARD

The CPU 6809 board includes the 6809 processor, its clock, a baud rate generator (with a separate 1.84 mHz crystal), and space for two banks of 4k PROM (or RAM). This on-board memory consists of two 2716 type sockets at $F000 and two at $F800 (dip switch selectable). A 4 mHz crystal is used to provide a 1 mHz E clock. The data buss drivers are 8835 devices compatible with the inverted nature of the SS-50 buss.

I installed my monitor at $F800 (more about that later) and a 6116 CMOS RAM at $F000 for some out-of-the-way RAM. Provisions are made to use the baud rate outputs (to the buss) as memory bank switching rather than baud clocks. Since the mother board also includes a baud-rate generator using the same circuit, I elected not to use the one on the CPU board, and as I had only one RAM board, I didn't implement the bank switching either. The board includes provisions for latching an external BREQ and HALT although there is no reason to do so on the board set I have.

Once I figured out what not to install, this board went together with no problems and worked the first time I turned it on.

VIDEO DISPLAY

The video display board provides a memory-mapped video terminal for the SS-50 buss using the 6845 video controller. Output is both composite video to directly drive a monitor and separate video and sync at TTL levels. A 12.576 mHz crystal (where do you get one of those?) supplies the dot clock. Included is a one-page memory (4-2116s) and space for a character set in a 2716 PROM. No provisions are made for inverted video or graphics nor does the board include a PIA for a keyboard interface. The design does not use any of the techniques to eliminate "spreckles" on the screen when both the CPU and the 6845 access the video memory.

The major problem I had with this board was getting the crystal to oscillate. I found that by replacing the 470 ohm resistor with a 1K and adding another 1K between pins 3 and 4 of the inverter, I could get reliable crystal starting. The circuit is then the same as Motorola uses on one of their video boards. There are already unmarked pads for the second 1K. Further, I found that there was a race condition with the 7404 inverter which was solved by replacing it with a 74LS04. The assembly documentation supplied with the board is mostly good except neither the schematic nor the assembly drawing had reference designators for the ICs (U-numbers). This sometimes made it difficult to find the right part without tracing the circuit on the board. The PC board itself did have references silkscreened on, but they didn't relate to anything.

Software supplied by DS68 for the video board include video drivers and a character set to be burned into a EPROM. The drivers were apparently copied from Motorola's CBUG and are intended to overlay tape routines in the SWTP SBUG. I used neither the driver nor the character set, electing to write my own.

Most video boards end up requiring dot clocks that mean, for the one-off builder, ordering a special crystal. Since I already had a 15 mHz crystal, I decided to try to use it. After struggling with the 6845 data sheet, I finally wrote a BASIC program to help set up the CRTC. I ended up with some non-standard horizontal and vertical frequencies (15,756 Hz and 50.02 Hz), but

they work well on my medium quality monitor. More on all the software later.

64K RAM

The 64K RAM board uses 4116 dynamic RAMS in the memory and 3242 and 3480 as controllers (both Motorola parts). Two delay lines are used to set up the various clocking patterns. The board is addressable in 4K blocks, allowing one to steer around the I/O and ROM above $E000. Also, it is possible to bank-switch for systems larger than 64K.

This board gave me the most trouble through no fault of DS68. I used RAM chips salvaged from many conversions of TRS-80Cs to 64K. This was a big mistake and held up the completion of the computer longer than I'll admit. Once I installed new RAM chips the board worked perfectly. BUY GOOD RAMS! When the RAM board is ordered, the delay lines should be ordered at the same time because they are special and not available anyplace else (that I could find).

DISK CONTROLLER FDC-50

The disk controller board is the most complicated of the boards available from DS68 and well it should be--it has a tough task to do. The design of the board closely follows Western Digital practice as described in their data sheets and app notes. A WD 1791 controller is used which allows both single- and double-density formats. Data separation is done by the preferred phase-lock-loop method. The controller will control up to four 5 1/4" drives, although the power supply is rated for only two. I am currently running three drives from the supply and haven't seen any smoke yet.

The board went together easily and, after correcting of what seemed to be a miswire of one control, worked well. Most circuits show R1 (the precomp control) as grounded on the bottom of the pot. This is easily fixed by a little trace cutting and a jumper wire.

Software supplied by DS68 for this card included source for driver routines, patches for TSC's NEWDISK.CMD and a BOOT routine (my manual was missing the BOOT). The drivers support single- and double-density and single- and double-sided drives. There is also partial capability to read 40 track disks on an 80 track drive. Unfortunately, there is no way to tell the routine that a 40 track disk is inserted. Barrie Smith, in the Feb. '84 BIT BUCKET, described a neat solution that works. My drives include a Shugart SA460 80 track double-sided which requires a time delay before switching sides. I fixed this with some additional logic in the SEEK routine.

The NEWDISK patches declare the format at assembly time (so does TSC's). This is OK if you have only one kind of drive. Since I have several, I adopted Steve Odneal's F-MATE(RS) patches that allow inputting format parameters at run time.

DS68 has a DMA board available that works in conjunction with the disk controller for faster data transfer but since I don't have the board, I can't report on it.

MULTI-I/O BOARD

The Multi-I/O card contains two 6850 ACIAs and two 6821 PIAs providing two serial ports and four 8-bit parallel ports. Buffering is provided for the ACIAs (by 1488s and 1489s) to RS-232 levels.

The baud clocks for the ACIAs is generated off the board, either on the CPU card or on the mother board, in the normal SS-50 buss manner. One of the PIA ports is expected to be used as a keyboard interface; the remaining three are available for whatever the user desires. None of the parallel ports are buffered. The board is addressable in the $E7E0-$E7F8 area by jumpers. NMI and IRQ access is also provided by jumpers.

The construction and operation of the I/O card is straightforward and no problems were experienced. I added a 74LS240 buffer to interface with my parallel printer and a counter circuit that is part of a tracer routine in my monitor. Both of these circuits were mounted to the card by double-backed tape and scramble wired.

DUAL SERIAL INTERFACE CARD

This card provides two serial ports with buffering to RS-232 levels and mounts in a SS-30 port on the mother board. The circuitry is the same as the serial ports of the Multi-I/O board described above. There is a switch driven flip-flop on this board that connects to the NMI line that can be used for single-stepping. To quote the manual, "..additional software is required".

Unless you require more then two serial ports, I'd recommend you not purchase this card. Not that there's anything wrong with it, just that the Multi-I/O has the same serial capability and you will need the parallel ports of the Multi-I/O for keyboard interface.

MOTHER BOARD

The Mother Board has sockets for eight 50-pin boards and eight 30-pin boards. Included on the board is a baud rate generator and address decoding for the 30-pin sockets. Decoding can be done in any 4K block, including the preferred $E000 area. A MC14411 and 1.8432 mHz crystal are used in the baud rate generator and provide all of the standard baud clocks from 110 Hz to 9.6 kHz. The 110, 300, 1200, 4800 and 9600 Hz outputs are currently buffered but this can be easily changed. The board is fabricated from 3/32" epoxy material and is quite sturdy.

POWER SUPPLY AND ENCLOSURE

DS68 does not sell either a power supply or a case for the computer; here you are on your own. The possibilities are to either buy from another SS-50 supplier or make your own. I chose the latter. For the power supply, I used a transformer intended for S-100 systems with bridge rectifiers and filter capacitors from my junk box. A LMB Uni-Pac (7"X17"X14") was used for the case. Be sure to include a fan (and air exit holes) if your case is closed. It gets hot in there.

ADDITIONAL SOFTWARE

As mentioned earlier, DS68 assumes that you own a copy of the General Version of FLEX and that you have a monitor such as SWTP SBUG E. Supplied with the various board manuals are source listings pertinent to that board. In my case, I did not have a monitor and I wanted to customize some of the DS68 software.

Software that I have written for the computer includes the following:

a) S-MON - a firmware monitor.
b) An enhancement of the disk driver - I/O routines provided by DS68.

c) A "NEWDATE" routine that keeps the current date on the BOOT so that the date doesn't have to be input every time on multiple boots on the same day.
d) An adaption of NEWDISK (called FORMAT) that allows formatting parameters to be declared at run time.
e) A CAT.CMD that combines TSC's CAT and a DIR .CMD described in '68' MicroJournal.
f) A BASIC program for setting up the MC6845.
g) An improved character set with "nicer" descenders for the video board.
h) An offset binary loader - like GET with an offset.

** A disk containing the source code for the above software has been supplied to the Journal. However, at this writing I don't if or how it will be made available. Some or all of the programs may be published. As for the monitor and character set, you will need access to an EPROM burner to use them. I may be talked into burning the EPROMS if it doesn't get to be a big hassle. Or maybe DS68 can be convinced to offer them as a product.

COST

Below is the approximate cost of my complete DS68 computer. I'm sure I left some things out, but the total is not too bad considering the result.

Circuit boards (DS68)	$325
SS-50 connectors and delay lines(DS68)	95
IC's (JDR Microdevices)	250
Additional IC's (Rams, etc.)	150
Connectors	50
Power Supply	50
Enclosure and Fan (LMB)	75
Video Monitor (Sanyo)	150
Keyboard	135
Approx. Total	$1280
Printer and Disk Drives (shop around)	?

IN CONCLUSION

Was it worth it? For me - yes. I have a computer that has all the features I need at a price I can live with. If, however, you are not comfortable with a schematic or a soldering iron, then this is not for you. Or if you have to pay for somebody's time to put it together, then there are better ways to go. But if you're into computers as a hobby and enjoy putting things together, then you should seriously consider the DS68 computer. SDL

A SPECIAL NOTE:

From time to time I have expounded upon my roots as an 'original' Standard S50 Bus computer hobbyists. I know many successful (and some rich) professionals who started as a hobbyists, with a SWTPC, S58, MSI, Altair or Sphere, and grew professionally from there. Betcha most of you never heard of Sphere, they sold 6800 kits also, in the dark ages.

I could spend hours reciting to you stories, mostly true, of hobbyists, I have had the pleasure of knowing, who were transformed practically instantly from pure hobbyists to skilled professional, and most could practically name their price. In those days hands on experience was the key attribute and acedemics were secondary. Actually in those days there were no micro ecademics, but there were a lot of hobbyists doing their thing, and that thing was exactly what the explosive micro industry was looking for. Now those days are now gone, forever. The woods are full of high class colleges and universities loaded to the gills with micro-computer related classes. Micro (experts?) are being turned out by the buckets full. But back then, when getting the job done right was the primary requirement, the hobbyists were the folks who got the ball rolling. I still pride myself as being a - hobbyists. I like to build. I get immense pleasure in watching the boards, parts, software (mostly home-brewed) and all the other stuff come together into a computer that I can rightly say about - "I BUILT IT!"

This brings me to this, the hobbyists has practically no place to turn nowadays, for 6809 kits and boards. The reasons are varied, but basically they are the old saw of economics. Too little profit and too much work, with a small marketplace. The last of the full time kit board houses stopped advertising 6809 kits and bare boards a few months back - Data Systems 68. The reason, (see four lines up)!

Now, it is well known that I have always had, and always will have a very special feeling about that group of us 'called' hobbyists. Without us the explosion would have struggled much harder in the beginning to really bang. Fact is without us it still might be in the early formative stages. A lot of engineering know-how was developed by 'us' debugging someone elses hardware or software. Wonder where they would be now without us at that time? However, I cannot fault anyone for leaving the hobby marketplace. It is a tough one. Answering what seems to be an endless stream of 'stupid' beginners (and some not beginners) whys, hows and wheres. Well, stupid it might have seemed but as I remember it, we all, at times asked the same questions. But for all the reasons, until now, the hobby and professionel or commercial kit and bare-board suppliers have about all grown away or faded from view. Leaving the guy, or gal, who has the desire, because of economics or pride of 'doing', without a one-stop source of the basic and fundamental building block - **bare boards that work!**

I have received more letters and telephone calls concerning the above subject, than any other, over the past years, as it got worse and worse. Go back a few years and compare how many companies, large and small, were selling bare boards and kits, with what is advertised today. Where did they all go? Well, most went down the tube. Some for poor business sense, some because of poor products (not many), some because the 'hand-holding' was just too much and there was more money to be made elsewhere, and some grew into bigger and better things. But no matter what the reason, the ultimate loser was the, right - hobbyists (or is it professional who needs to modify a particular board function), I really don't know, but I do know that hundreds of you have lamented, to me, the passing of kits and good bare boards.

This past week I have entered into an agreement with Data System 68 to exclusively distribute their boards and other hobby-kit products to our readers. In an attempt to avoid some of the problems that caused them to leave the market, some slight price adjustments will be made over what was advertised in their final ads. Most items however, will be sold at the last advertised prices. **Since I started 68 Micro Journal, over 6 years ago, I have made little if any money from it. But due to other business considerations we have made ends meet and staved off the wolf. Now, with your help, if I understand what you have been telling me, I will stick with this for as long as I don't lose our corporate shirt!** But you gotta help.

Agreed that the quality of these boards are above average and that the price is certainly right, and even if you all buy a lot, it still might not work. We can only answer so many questions and hold so many hands. While our support will be good I feel that you should realize that it does take some level of skill to do the whole thing right. If you have NEVER soldered a wire or put a kit or board together, well, you might have some problems. I know many, many who built kits that worked

the first time around, but some didn't. However, for the not-feign of heart, I suspect you will find it a rewarding and enjoyable adventure. The knowledge gained by trouble shooting your mistakes (and possibly ours, although I hope not) and twiddling the software to your liking to develop your very own, built by me, computer, that works as well as those 'store bought ready rolled' ones, is a thrill that only the doing can bring about.

DMW

- - -

Another Editors' Note: Since the above was written there has been a nice change. I am NOT going to be in the board business - we were able to interest **DIGITAL RESEARCH COMPUTER (of Texas)** into buying out the entire line of boards. As most of you know what a swell job they have done with boards and kit, even to a full 64K 6809 Computer kit. My main concern is fulfilled; that is that a complete line of bare boards be available to you who still like or need to 'roll your own'. I am always glad not to get into anything that conflicts with my other advertisers. BUT, if someone does NOT do it, I will. In this case it turns out fine.

Thanks Jim, looking forward to your new line of 6809 computer boards.

DMW

REMOTE ANALOG TO DIGITAL CONVERSION

REMOTE ANALOG TO DIGITAL CONVERSION

An MC14469 addressable asynchronous receiver/transmitter interfaced to an ADC0817 analog to digital converter as shown in the accompanying schematics. Control of the AART and A/D converter was effected by a serial interface card on PORT 4. Sixteen channels of data can be sampled and received by a remote computer using transmit, receive and ground lines. In addition to this, another eight bit word can be sampled by the AART. Since the AART is hardware programmed to respond to a particular address which the computer sends out, as many as 128 AART's can be tied to the transmit, receive and ground lines.

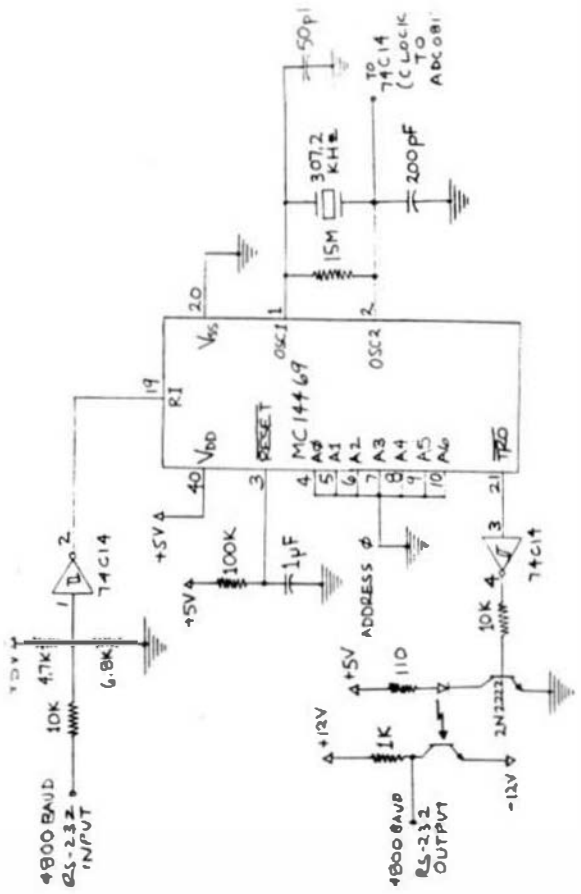
Once the MC6850 of the serial interface card has been initialized to transmit and receive in a 8 bit word, even parity, and one stop bit format, a unique address is sent out. Command words are then sent to the AART to start the A/D conversion for the channel selected. When the A/D conversion is complete, the end of conversion pulse causes the AART to send two words of data to the serial card. The second word is the eight bit value of the voltage for the channel selected. The eight bits of the first word and three bits of the command word are available for other uses. The AART communicates back and forth with the serial card at 4800 baud. A 307.2 Khz ceramic resonator was obtained from Radio Materials Company, 4242 N. Bryn Mawr Avenue, Chicago IL 60646. Their minimum order is $150.00. An optoisolator was used to translate the 0 to +5 volt excursions of the AART to +12 and -12 volt levels. The collector to emitter voltage of the optoisolator should be rated at about 40 volts. A 5 volt power supply with a DC/DC converter can be used to provide the +12 and -12 volts. Otherwise, in addition to the 5 volt supply, a +12 and -12 volt supply must be provided.

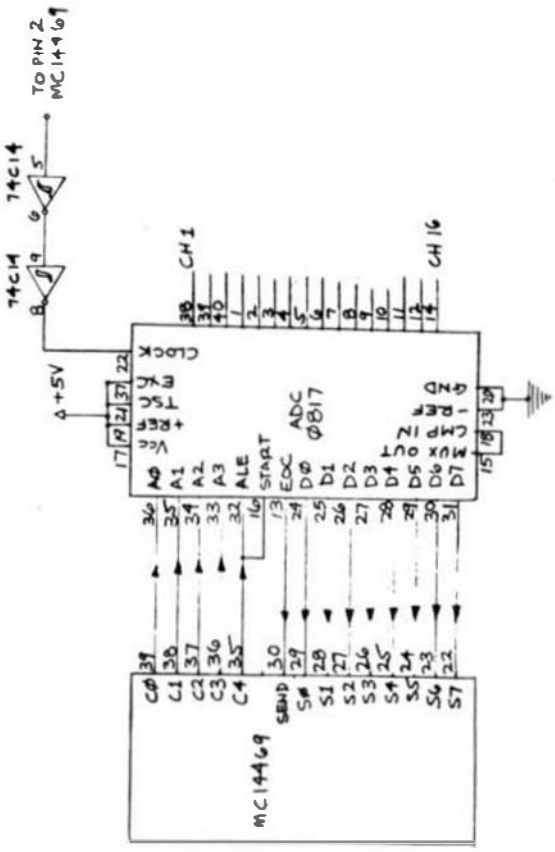
The BASIC program REMOTE calls a USR routine which samples all sixteen channels of the A/D converter. The value of the first eight bit word is reported also. The assembly language program REATODM samples all sixteen channels of the A/D converter as well as the first eight bit word.

Jeffrey M. Craig
Apt. 912 - 3001 S. King Dr.
Chicago, IL 60616
21 August 1982

```
1.00=
2.00=* THIS PROGRAM USES FLEX 2 ROUTINES
3.00=* THIS PROGRAM USES AN MC14469 ADDRESSABLE ASYNCHRONOUS RECEIVER
4.00=* TRANSMITTER TO CONTROL A REMOTE ANALOG TO DIGITAL CONVERTER
5.00=* VIA A SERIAL INTERFACE
6.00=
7.00= NAM REATODM
8.00= OPT PAG
9.00= ORG $1000
10.00=
11.00=DATA EQU $8011
12.00=CONTRL EQU $8010
13.00=PCRLF EQU $AD74
14.00=OUTHEX EQU $AD3C
15.00=WARM EQU $AD03
16.00=MEM EQU $2000
17.00=OUTCH EQU $AD9F
18.00=
19.00=* CONFIGURE 6850 TO DIVIDE INCOMING DATA BY 16 CLOCK
20.00=* CONFIGURE 6850 FOR 8 BITS,EVEN PARITY,ONE STOP BIT
21.00=BEGIN LDA A #%00011001
22.00= STA A CONTRL
23.00=* CHECK TO SEE IF TRANSMIT DATA REGISTER IS EMPTY
24.00= JSR LOOP1
25.00= LDA B #16
26.00=* CALL UP THE AART
27.00=START LDA A #%10000000
28.00= STA A DATA
29.00=* CHECK TO SEE IF TRANSMIT DATA REGISTER IS EMPTY
30.00= JSR LOOP1
31.00=* INITIALIZE THE ANALOG TO DIGITAL CONVERTER
32.00= TBA
33.00= SUB A #16
34.00= STA A DATA
35.00=* CHECK TO SEE IF TRANSMIT DATA REGISTER IS EMPTY
36.00= JSR LOOP1
37.00=* SELECT CHANNEL
38.00= TBA
39.00= STA A DATA
40.00=* CHECK TO SEE IF TRANSMIT DATA REGISTER IS EMPTY
41.00= JSR LOOP1
42.00=* START ANALOG TO DIGITAL CONVERSION
43.00= TBA
44.00= SUB A #16
45.00= STA A DATA
46.00=* CHECK TO SEE IF TRANSMIT DATA REGISTER IS EMPTY
47.00= JSR LOOP1
48.00=* CHECK TO SEE IF RECEIVE DATA REGISTER IS FULL
49.00= JSR LOOP2
50.00=* GET DATA AND DISPLAY IT
51.00= LDA A DATA
52.00= STA A MEM
53.00= JSR PCRLF
54.00= LDX #MEM
55.00= JSR OUTHEX
56.00= LDA A #$20
57.00= JSR OUTCH
58.00= JSR LOOP2
59.00= LDA A DATA
60.00= STA A MEM
61.00= LDX #MEM
62.00= JSR OUTHEX
63.00= LDA A #$20
64.00= JSR OUTCH
65.00= INC B
66.00= CMP B #32
67.00= BNE START
```

```
68.00= JMP WARM
69.00=LOOP1 LDA A CONTRL
70.00= AND A #%00000010
71.00= CMP A #%00000010
72.00= BNE LOOP1
73.00= RTS
74.00=LOOP2 LDA A CONTRL
75.00= AND A #%00000001
76.00= CMP A #%00000001
77.00= BNE LOOP2
78.00= RTS
79.00= END BEGIN
 1.00=
 2.00=* THIS PROGRAM IS A USER ROUTINE FOR BASIC
 3.00=* THIS PROGRAM USES AN MC14469 ADDRESSABLE ASYNCHRONOUS RECEIVER
 4.00=* TRANSMITTER TO CONTROL A REMOTE ANALOG TO DIGITAL CONVERTER
 5.00=* VIA A SERIAL INTERFACE
 6.00=
 7.00= WAM READ0
 8.00= OPT PAG
 9.00= ORG $6000
10.00=
11.00=DATA EQU $8011
12.00=CONTRL EQU $8010
13.00=
14.00=* CONFIGURE 6850 TO DIVIDE INCOMING DATA BY 16 CLOCK
15.00=* CONFIGURE 6850 FOR 8 BITS,EVEN PARITY,ONE STOP BIT
16.00=BEGIN LDA A #%00001001
17.00= STA A CONTRL
18.00=* CHECK TO SEE IF TRANSMIT DATA REGISTER IS EMPTY
19.00= JSR LOOP1
20.00= LDX #$7000
21.00= LDA B #16
22.00=* CALL UP THE AART
23.00=START LDA A #%10000000
24.00= STA A DATA
25.00=* CHECK TO SEE IF TRANSMIT DATA REGISTER IS EMPTY
26.00= JSR LOOP1
27.00=* INITIALIZE THE ANALOG TO DIGITAL CONVERTER
28.00= TBA
29.00= SUB A #16
30.00= STA A DATA
31.00=* CHECK TO SEE IF TRANSMIT DATA REGISTER IS EMPTY
32.00= JSR LOOP1
33.00=* SELECT CHANNEL
34.00= TBA
35.00= STA A DATA
36.00=* CHECK TO SEE IF TRANSMIT DATA REGISTER IS EMPTY
37.00= JSR LOOP1
38.00=* START ANALOG TO DIGITAL CONVERSION
39.00= TBA
40.00= SUB A #16
41.00= STA A DATA
42.00=* CHECK TO SEE IF TRANSMIT DATA REGISTER IS EMPTY
43.00= JSR LOOP1
44.00=* CHECK TO SEE IF RECEIVE DATA REGISTER IS FULL
45.00= JSR LOOP2
46.00=* GET DATA AND DISPLAY IT
47.00= LDA A DATA
48.00= STA A 0,X
49.00= INX
50.00= JSR LOOP2
51.00= LDA A DATA
52.00= STA A 0,X
53.00= INX
54.00= INC B
55.00= CMP B #32
56.00= BNE START
57.00= RTS
58.00=LOOP1 LDA A CONTRL
59.00= AND A #%00000010
60.00= CMP A #%00000010
61.00= BNE LOOP1
62.00= RTS
63.00=LOOP2 LDA A CONTRL
64.00= AND A #%00000001
65.00= CMP A #%00000001
66.00= BNE LOOP2
67.00= RTS
68.00= END BEGIN
```





6809 FLEX DISKETTE INVENTORY

It is easy to acquire diskettes containing potentially useful files. For safety reasons, it is best to copy these files onto other diskettes. However, it is quite often hard to remember which diskettes contain backup copies of each file. The solution to this problem is to create an inventory of all files indicating how many copies of each file exist and which diskettes contain the copies. This article describes a program to generate just such an inventory.

The program makes several assumptions as follows. You must have a 6809 microcomputer using the FLEX™ disk operating system and at least two disk drives. You must have the TSC Sort/Merge software package. The diskettes to be inventoried must be in FLEX™ format and should have unique volume numbers, but all may have the same volume name. If you do not have 2K bytes of memory mapped to start at address A000 or do not have a serial printer on port 7, simple changes must be made to the program as described later.

The program is easy to use and is shown in Listing 1. Place in drive 0 the master program diskette containing the following files: "index.cmd", "sortspec.bin", "S.CMD", "PSORT.CMD", "SRTMRG.SYS", and "LIST.CMD". The command "0.index" causes the process to begin. A prompt is given to insert into drive 1 a diskette containing plenty of free space. Several work files will be placed on this diskette. This diskette must not be removed until all processing is completed. These work files will be deleted leaving only the file "diskindx.txt" which contains the final inventory. Soon the prompt "Insert disk in drive 0 and press return. Press "S" to stop." will appear. Each diskette to be inventoried is inserted in turn into drive 0 and the return key pressed. The master program diskette may be included in the inventory, but the work diskette (in drive 1) may not be included. If the directory of any diskette cannot be read, a prompt will appear to remove the diskette and reinsert it. After all directories have been read, press the "S" key in response to the prompt for another diskette. Either upper or lower case "S" will work. Reinsert the master program diskette into drive 0 and press the return key in response to the prompt. The TSC Sort/Merge program will sort the files into alphabetic order by the file name extension and file name. (I.e., the CMD files will come before the TXT files.) A prompt will then appear requesting a title of up to 40 characters for the inventory. The date of the inventory and the volume name (if common to all the diskettes) are appropriate as part of the title. This title will appear at the top of each page of the inventory. After a pause to combine multiple occurrences of the same file into a single entry and create the final inventory, the final inventory will be printed on a serial printer on port 7. If the printer is attached through port 0, change the "0.S" in the familiar FLEX command portion of line number 373 in the listing to "0.S#0". If a parallel printer is attached to port 7, change the "0.S" to "0.P" and replace the file "S.CMD" on the master program diskette by the file "P.CMD". FLEX entry points may need to be changed for eight inch diskettes. Other minor changes may be needed to adapt the program to your system.

A quick look through the program will show its basic organization. Lines 14 through 28 simply permit meaningful names to be used in place of certain constants. Lines 32 through 39 issue the prompt to insert the work diskette in drive 1 and initialize the first work file. (The format of each record in the work file is described in lines 340 through 346.) Lines 41 through 86 issue the prompt to insert the next diskette to be indexed in drive 0 (41-47), read the volume number (48-59), read each directory entry (66-76), and write the work file records (77-84). After all information has been extracted from the diskette, lines 89 through 109 close the first work file (89-92), call the FLEX sort routine to sort the records into a second work file (same format) (93-105), and then delete the first work file (106-109). Lines 110 through 127 prompt for a title for the final output.

Backspace (cntl H) and cancel (cntl X) are recognized. Lines 128 through 144 prepare the files on the work diskette. Lines 145 through 215 remove duplicate file names and format the final output. (The format of the final output is described in lines 350 through 359.) This process is complicated by the fact that upper and lower case file names are considered equivalent. The case of each character in the file name is taken from the first occurrence of the file. For example, in the sample output, diskette 14 contained "LIST.CMD". Diskettes 19 and 33 contained "list.cmd" and diskette 27 contained "list.CMD". All these are considered equivalent. When all information from the sorted work file has been processed, lines 217 through 236 output the final output records and delete the sorted work file. Lines 237 through 245 cause the final output to be printed and return control to the operating system. Several minor subroutines follow. The subroutine in lines 261 through 267 outputs the header line for the top of each page. Lines 269 through 276 output a record to the final file. The subroutine in lines 282 through 320 converts 16-bit binary numbers to five-character ASCII numbers with leading blanks, if necessary. Lines 362 through 365 contain variables used by the program. Lines 368 through 371 contain file names needed to redirect the FCBs. Work file names are purposely made odd to avoid any possible duplication. Lines 373 through 376 are the FLEX commands used by the program. Lines 378 through 391 are the prompts to the user. Lines 394 and 395 are the construction area for the final output record. Lines 379 through 400 contain the top of page header. Lines 402 and 403 are the input FCB. Lines 405 through 409 are the overlapped output FCB.

A sort specification file named "sortspec.bin" must also be on the master program disk. The easiest way to create this file is to use the TSC SORT command. This program will issue a series of prompts for sort parameters. The exact prompts probably depend upon the version of the TSC sort package you have, but they should look something like those in TABLE I. You should respond to each prompt as shown in TABLE I. Each response is terminated by a carriage return, and some responses are only a carriage return.

In short, this program fills a need for anyone experiencing difficulty keeping track of diskette files. This program is straightforward and can be easily modified to meet the system configuration of any 6809 FLEX user having two or more disk drives. If you do not wish to input the program yourself, I will send you a FLEX formatted minidiskette containing the program (in both text and command forms) and the "sortspec.bin" file for $5.

FLEX™ is a registered trademark of Technical Systems Consultants, Inc., Layfayette, Indiana.

TABLE I
SORTSPEC.BIN File Creation Prompts

```
output to disk?  y
filename?  1.Q-XZVJGW.QHB
intermediate work file drive?  1
fixed or variable length records?
EOR character?
field separator character?
output from key, input, or other?
input keys
?  9-11,1-8,17-21
?
output keys
?
further options?  y
input file text or binary?
alternate collating sequence?
lower case equivalent to upper?  y
delete records with blank keys?
select/exclude option?
output text file or binary?
print messages?
save parameters?  y
filename?  0.SORTSPEC.BIN
exit or proceed?  e
```

```
5  A000                 org   $a000
6                       *
7                       * INDEX program to generate file index for diskettes
8                       *
9                       * copyright 1981 by Tom Weaver
10                      *                       825 W. Sherry
11                      *                       Normal, Ohio 73069
12                      *                       (405) 364-6856
13                      *
14              C080    linbuf  equ   $c080     flex9 line buffer
15              CC14    bufpnt  equ   $cc14     flex9 line buffer pointer
16              CD03    warms   equ   $cd03     flex9 warm start address
17              CD0C    inch    eq    $cd0c     flex9 terminal character input
18              CD1E    pdata   equ   $cd1e     flex9 terminal string output
19              CD3F    rpterr  equ   $cd3f     flex9 error report
20              CD4B    flex9a  equ   $cd4b     flex9 as a subroutine entry point
21              DA06    fms     equ   $da06     file management system entry point
22              0001    opnin   equ   1
23              0002    opnout  equ   2
24              0004    close   equ   4
25              0006    opndir  equ   6
26              0007    getdir  equ   7
27              000C    delete  equ   12
28              0010    opninf  equ   16
29                      *
30                      *
31                      *
32  A000 8E   A329      ind8a   ldx   #remark
```

```
A003 BD CD1E          jsr   pdsin    give prompt for drive 1
A006 BD CD0C          jsr   inch     wait until ready
A009 8E A68C          ldx   #tmpfil
A00C BD D406          jsr   fms      delete any garbage
A00F 8E A6AD          ldx   #filout
A012 BD D406          jsr   fms      open initial work file
A015 1026 2D26        lbne  rpterr
                      *
A019 8E 83BF  mainlp  ldx   #prompt
A01C BD CD1E          jsr   pdstr    output prompt for new disk
A01F BD CD0C          jsr   inch     input reply
A022 81 53            cmpa  #'S      stop ?
A024 27 68            beq   done     yes
A026 81 73            cmpa  #'s      stop ?
A028 27 64            beq   done     yes
A02A 8E A56D          ldx   #filin
A02D 86 10            lda   #opninf
A02F A7 84            sta   ,x
A031 BD D406          jsr   fms      open sys info rec
A034 1026 01C1        lbne  error    disk read error - attempt recovery
A038 86 07            lda   #getdir
A03A A7 84            sta   ,x
A03C BD D406          jsr   fms      read sys info record
A03F 102 0186         lbne  error    disk read error - attempt recovery

A043 AE 0F            ldx   15,x     disk number
A045 CE A285          ldu   #dlebac
A048 BD A23C          jsr   cvtnum   convert binary to ASCII
A04B 8E A56D          ldx   #filin
A04E 86 06            lda   #opndir
A050 A7 84            sta   ,x
A052 BD D406          jsr   fms      open directory
A055 1026 0140        lbne  error    disk read error - attempt recovery
                      *
A059 86 07    inloop  lda   #getdir
A05B A7 84            sta   ,x
A05D BD D406          jsr   fms      read directory entry
A060 27 09            beq   inok     good error code
A062 A5 01            lda   1,x      end of file ?
A064 81 03            cmpa  #3
A066 27 B1            beq   mainlp   yes
A068 7E A1F9          jmp   error    no - attempt recovery
A06B 6D 0A    inok    tst   8,x      directory entry used ?
A06D 27 4A            beq   mainlp   no - erase
A06F 2B E8            bmi   inloop   yes - deleted
A071 CE A2A0          ldu   #size
A074 8E A582          ldx   #line21  size of file
A077 BD A23C          jsr   cvtnum   convert binary to ASCII
A07A CE A571          ldu   #filine1
A07D 8E A6AD          ldx   #filout
A080 BD A1A1          jsr   outlit   output file name
A083 CE A2AD          ldu   #size    jsr outlit
A086 BD A1A1          jsr   outlit   output file size and disk number
A089 8E A56D          ldx   #filin
A08C 20 CB            bra   inloop   go read next entry for this disk
                      *
                      *
A08E 8E A6AD  done    ldx   #filout
A091 86 04            lda   #close
A093 A7 84            sta   ,x
A095 BD D406          jsr   fms      close output file
A098 8E A35B          ldx   #zapler
A09B BD CD1E          jsr   pdstr    get DOS disk back in drive 0
A09E BD CD0C          jsr   inch
A0A1 CE C080          ldu   #linbuf  flex9 line buffer
A0A4 8E A2FA          ldx   #srtcsi
A0A7 BF CC14          stx   bufpnt   flex9 line buffer pointer
A0AA EC 81    callop  ldd   ,x++
A0AC ED C1            std   ,u++
A0AE C1 0D            cmpb  #$0d     carriage return ?
A0B0 26 F8            bne   callop
A0B2 BD CD4B          jsr   flex9a   call flex sort as a subroutine
A0B5 5D               tstb           everything ok ?
A0B6 1026 0179        lbne  badsrt   no
A0BA 86 0C            lda   #delete
A0BC 8E A6AD          ldx   #filout
A0BF A7 84            sta   ,x
A0C1 BD D406          jsr   fms      delete initial work file
A0C4 8E A3FE  asktit  ldx   #title   prompt for title/date
A0C7 BD CD1E          jsr   pdstr
A0CA 8E A4ED          ldx   #head/2
A0CD C6 50            ldb   #80      maximum title length
A0CF BD CD0C  titlop  jsr   inch     input char from terminal
A0D2 81 08            cmpa  #$08     backspace ?
A0D4 1027 012A        lbeq  bkchup   yes
A0D8 81 0D            cmpa  #$0d     return
A0DA 27 09            beq   titdun   yes - end of title
A0DC 81 18            cmpa  #$18     line cancel ?
A0DE 27 E8            beq   asktit   yes - start title over
A0E0 A7 80            sta   ,x+      save character
A0E2 5A               decb           title too long ?
A0E3 26 EA            bne   titlop   no - get more
A0E5 CC 0D0D  titdun  ldd   #$0d0d
A0E8 ED 81            std   ,x++
A0EA 86 04            lda   #4
A0EC A7 84            sta   ,x
A0EE 8E A570          ldx   #linnum-1
A0F1 CE A2AB          ldu   #sorted
A0F4 BD A294          jsr   movnam   name of sorted work file
A0F7 8E A680          ldx   #outnam-1
A0FA CE A2B7          ldu   #dskid+
A0FD BD A294          jsr   movnam   name of final file
A100 8E A6AD          ldx   #filout
A103 86 0C            lda   #delete
A105 A7 84            sta   ,x
A107 BD D406          jsr   fms      delete any old INDEX
A10A 8E A680          ldx   #outnam-1
A10D CE A2B7          ldu   #dskid+
A110 BD A294          jsr   movnam   name of final file
A113 8E A AD          ldx   #filout
A116 86 02            lda   #opnout
A118 A7 84            sta   ,x
A11A BD D406          jsr   fms      open output file
A11D BD A211          jsr   phdr     output top of page header
A120 86 01            lda   #opnin
A122 A7 84            sta   ,x
A124 BD D406          jsr   fms      open sorted work file
A127 C6 0B            ldb   #8
A129 CE A422          ldu   #line+10
A12C BD A208  finnam  jsr   indata   read file name
A12F CE A42B          ldu   #line+19
A132 C6 0D            ldb   #13
A134 BD A208  finish  jsr   indata   read file name ext, size, and disk numb
A137 1F 32            tfr   u,y      save end of line pointer
A139 BD D406          jsr   fms      read carriage return
A13C CE A422          ldu   #line+10   point to start of file name
A13F BD D406  dupfin  jsr   fms      read char of new file name
A142 26 6C            bne   atend    must be end of file
A144 A7 CB EA         sta   -22,u    save this character
A147 A1 C0            cmpa  ,u+      does it match prev file name char ?
A149 26 1B            bne   notsam   no

A14B 5C       dupinc  incb
A14C C1 08            cmpb  #8
A14E 27 12            beq   ovrdot   skip over period
A150 C1 0B            cmpb  #11
A152 26 EB            bne   dupfin   check rest of name
A154 C6 05            ldb   #5
A156 BD D406  dupgit  jsr   fms      ignore file size
A159 5A               decb
A15A 26 FA            bne   dupgit
A15C 1F 23            tfr   y,u      end of line pointer
A15E C6 05            ldb   #5       just append disk number
A160 20 D2            bra   finish
                      *
A162 33 41    ovrdot  leau  1,u      skip over period
A164 20 D9            bra   dupfin
                      *
A166 A8 20    notsam  eora  #$20     check upper/lower case
A168 A1 C2            cmpa  -,u      match now ?
A16A 26 04            bne   newnam   no
A16C 33 41            leau  1,u      yes - continue comparison
A16E 20 0B            bra   dupinc
                      *
A170 3A       newnam  pshs  b,u      save current position
A172 CC 0D0A          ldd   #$0d0a
A175 ED 64            std   ,y
A177 CE A418          ldu   #line
A17A BD A221          jsr   outlin   output file info
A17D 7A A29E          dec   curlin   number lines left on this page
A180 26 03            bne   newlin
A182 BD A216          jsr   endpag   skip lines and print header
A185 CE A422  newlin  ldu   #line+10
A188 C6 0C            ldb   #12
A18A A6 CB E4 newfil  lda   -22,u    copy name of new file
A18D A7 C0            sta   ,u+
A18F 5A               decb
A190 26 F8            bne   newfil
A192 35 88            puls  b,u      restore pointers
A194 33 41            leau  1,u
A196 50               negb
A197 CB 07            addb  #7       length of name to finish
A199 27 98            beq   finish-5
A19B 2A 8F            bpl   finnam
A19D CB 0D            addb  #13
A19F 20 93            bra   finish
                      *
A1A1 C6 0B    outlit  ldb   #11      process 11 characters
A1A3 A6 C0    outchr  lda   ,u+      pick up character
A1A5 26 02            bne   nonull
A1A7 86 20            lda   #$20     replace null by blank
A1A9 BD D406  nonull  jsr   fms      write to disk file
A1AC 5A               decb           number of char to go
A1AD 26 F4            bne   outchr
A1AF 39               rts
                      *
A1B0 CC 0D0A  atend   ldd   #$0d0a
A1B3 ED A1            std   ,y++
A1B5 CE A418          ldu   #line
>A1B8 BD A221         jsr   outlin   output last line of file info
A1BB F6 A29E          ldb   curlin
A1BE CB 0B            addb  #11
A1C0 8E A6AD          ldx   #filout
A1C3 86 0D    endskp  lda   #$0d
A1C5 BD D406          jsr   fms      space to end of page
A1C8 5A               decb
A1C9 26 F8            bne   endskp
A1CB 86 04            lda   #close
A1CD A7 84            sta   ,x
A1CF B7 A56D          sta   filin
A1D2 BD D406          jsr   fms      close final output file
A1D5 8E A56D          ldx   #filin
A1D8 BD D406          jsr   fms      close sorted work file
A1DB 86 0C            lda   #delete
A1DD A7 84            sta   ,x
A1DF BD D406          jsr   fms      delete sorted work file
A1E2 CE C080          ldu   #linbuf  flex9 line buffer
A1E5 8E A2C3          ldx   #printt
A1E8 BF CC14          stx   bufpnt   flex9 line buffer pointer
A1EB EC 81    endlop  ldd   ,x++
A1ED ED C1            std   ,u++
A1EF 81 0D            cmpa  #$0d     return ?
A1F1 26 F8            bne   endlop
A1F3 BD CD4B          jsr   flex9a   call flex to print index
A1F6 7E CD03          jmp   warms    finished
                      *
A1F9 8E A392  error   ldx   #errmsg
A1FC BD CD1E          jsr   pdstr    disk read error message
A1FF 7E A019          jmp   mainlp   try it again
                      *
A202 30 1F    backup  leax  -1,x     backspace one position
A204 5C               incb
A205 7E A0CF          jmp   titlop
                      *
A208 BD D406  indata  jsr   fms      read a char from disk
A20B A7 C0            sta   ,u+
A20D 5A               decb
A20E 26 F8            bne   indata   go read more
A210 39               rts
                      *
A211 CE A4C7  phdr    ldu   #header   top of page header
A214 20 03            bra   *+5
A216 CE A4BC  endpag  ldu   #skip     don't print on perforation
A219 BD D406          jsr   outlin
A21B 86 35            lda   #53      number of lines left per page
A21D B7 A29E          sta   curlin
A220 39               rts
                      *
A221 8E A6AD  outlin  ldx   #filout   output FCB address
A224 A6 C0            lda   ,u+
A226 81 04            cmpa  #4       end of line ?
A228 27 05            beq   outdun   yes
A22A BD D406          jsr   fms      output a character
A22D 20 F5            bra   outlin+3
A22F 8E A56D  outdun  ldx   #filin   input FCB address
A232 39               rts
                      *
A233 8E A30E  badsrt  ldx   #sorted
A236 BD CD1E          jsr   pdstr    sort error message
A239 7E CD03          jmp   warms    give up
                      *
A23C 4F       ostnum  clra           turn off significance
A23D B7 A29F          sta   filler
A240 C6 30            ldb   #'0
A242 8C 2710  cvtlp   cmpx  #10000   over ten thousand ?
A245 2D 0C            blt   cvtthn   no
A247 F7 A29F          stb   filler   turn on significance
A24A 8A 30            ora   #'0      force to print
A24C 30 89 0BF0       leax  -10000,x subtract ten thousand
A250 4C               inca
A251 20 8F            bra   cvtlp
A253 A7 C4    cvtthn  sta   ,u       store ten thousands digit
```

```
A255 B6    A29F            lda    filler
A258 8C    03E8   cvt12    cmpx   #1000     over one thousand
A25B 2D    0C              blt    cvthun    no
A25D F7    A29F            stb    filler    turn on significance
A260 8A    30              ora    #'0       force to print
A262 30    89 FC18         leax   -1000,x   subtract a thousand
A266 4C                    inca
A267 20    EF              bra    cvt12
A269 A7    41     cvthun   sta    1,u       store thousands digit
A26B B6    A29F            lda    filler
A26E 8C    0064   cvt13    cmpx   #100      over a hundred ?
A271 2D    0B              blt    cvtten    no
A273 F7    A29F            stb    filler    turn on significance
A276 8A    30              ora    #'0       force to print
A278 30    88 9C           leax   -100,x    subtract a hundred
A27B 4C                    inca
A27C 20    F0              bra    cvt13
A27E A7    42     cvtten   sta    2,u       store hundreds digit
A280 B6    A29F            lda    filler
A283 8C    000A   cvt14    cmpx   #10       over ten ?
A286 2D    07              blt    cvtone    no
A288 8A    30              ora    #'0       force to print
A28A 30    16              leax   -10,x     subtract ten
A28C 4C                    inca
A28D 20    F4              bra    cvt14
A28F 30    8B     cvtone   leax   d,x       combine digits and force units to print
A291 AF    43              stx    3,u       store tens and units digits
A293 39                    rts
                  *
A294 C6    0C     movnam   ldb    #12       move 12 characters
A296 A6    C0              lda    ,u+
A298 A7    80              sta    ,x+
A29A 5A                    decb
A29B 26    F9              bne    movnam+2  go move more
A29D 39                    rts


                  *
                  * FCB directory offsets
                  *
                  *   0    -function code
                  *   3    -drive number
                  *  4-11  -file name
                  * 12-14  -file extension
                  * 21-22  -file size
                  *
                  * - - - - - - - - - - - - - - - - - - - - -
                  *
                  * work file format
                  *
                  *  1-8   -file name
                  *  9-11  -file extension
                  * 12-16  -file size
                  * 17-21  -disk number
                  *  22    -carriage return
                  *
                  * - - - - - - - - - - - - - - - - - - - - -
                  *
                  * final output file format
                  *
                  * 11-18  -file name
                  *  19    -period
                  * 20-22  -file extension
                  * 23-27  -file size
                  * 28-32  -first disk number
                  * 33-37  -second disk number
                  *  ...   -additional disk numbers
                  *   ?    -carriage return
                  *
                           pag

A29E 00           curlin   fcb    0
A29F 00           filler   fcb    0
A2A0              size     rmb    5
A2A5              diskno   rmb    5
A2AA 0D                    fcb    $0d
                  *
A2AB 01           sorted   fcb    1
A2AC 51 20 58 58           fcc    ' XZYJGW.QHB'
A2B7 01           dskids   fcb    1
A2B8 44 49 53 4B           fcc    'DISKINDXTXT'
                  *
A2C3 30 2E 53 20  printt   fcc    '0.S 0.LIST 1.DISKINDX'
A2D8 0D 0D                 fcb    $0d,$0d
A2DA 30 2E 50 53  srtcal   fcc    '0.PSORT 0.SORTSPEC 11.0-XZYJGW.QHBI 1.Z-X--XZQ.V-J'
A30C 0D 0D                 fcb    $0d,$0d
                  *
A30E 53 6F 72 74  srtbad   fcc    'Sort bombed - index aborted.'
A32A 04                    fcb    4
A32B 49 6E 73 65  remark   fcc    'Insert clean disk in drive 1 and press return. '
A35A 04                    fcb    4
A35B 49 6E 73 65  master   fcc    'Insert program disk back in drive 0 '
A37F 61 6E 64 20           fcc    'and press return. '
A391 04                    fcb    4
A392 44 69 73 6B  errmsg   fcc    'Disk read error.  Reset disk and try again.'
A3BE 04                    fcb    4
A3BF 49 6E 73 65  prompt   fcc    'Insert disk in drive 0 and press return.  '
A3E9 45 6E 74 65           fcc    'Enter "S" to stop:  '
A3FD 04                    fcb    4
A3FE 44 61 74 65  title    fcc    'Date/Title:  '
A40B 04                    fcb    4
                  *
A40C 20 20 20 20           fcc    '
A418 20 20 20 20  line     fcc    '
A43A                       rmb    130
                  *
A4BC 0D 0D 0D 0D  skip     fcb    $0d,$0d,$0d,$0d,$0d,$0d,$0d,$0d,$0d,$0d,$0d
A4C7 20 20 20 20  header   fcc    '              File name    Size    Disks    '
A4ED 20 20 20 20  headr2   fcc    '                                            '
A515 20 20 20 20           fcc    '                                            '
A541 20 20 20 20           fcc    '                                            '
                  *
A56D 06 00 00 00  filin    fcb    opndir,0,0,0
A571              innam    rmb    316
                  *
A6AD 02 00 00 01  filout   fcb    opnout,0,0,1
A6B1 5A 2D 58 2D  outnam   fcc    'Z-X--XZQV-J'
A6BC 0C 00 00 01  tmpfil   fcb    delete,0,0,1
A6C0 5A 2D 58 2D           fcc    'Z-X--XZQV-J'
A6CB                       rmb    305
                  *
                           end    index
```

SYMBOL TABLE:

```
asktit A0C4   atcod  A1B0   backup A202   bedart A233   bufpnt CC1B
calltop A0A4  close  000s   curlin A29E   cvthun A269   cvt11  A242
cvt12  A258   cvt13  A26E   cvt14  A283   cvtnum A23C   cvtone A28F
cvtten A27E   cvttho A253   delete 000C   diskno A2A5   done   A08E
dskids A2B7   dupfin A13F   dupinc A14B   duppit A196   endlop A1EB
endpag A210   endskp A1C3   errmsg A392   error  A1F9   filin  A56D
filler A29F   filout A6AD   finish A134   finnam A12C   flex$s CD4B
fms    D406   getdir 0007   header A4C7   headr2 A4ED   iaab   CD0C
indata A208   index  A000   inloop A059   innam  A571   look   A068
linbuf C080   line   A418   mainlp A019   master A35B   movnam A294
newfil A18A   newlin A185   nowrun A170   nomult A189   notasm A104
opndir 0006   opnin  0001   opninf 0010   opnout 0002   outchr A143
outdun A22F   outit  A1A1   outlin A221   outnam A6B1   ovrdot A162
pdata  CD1E   phdr   A211   printt A2C3   prompt A3BF   remark A32B
rpterr CD3F   size   A2A0   skip   A4BC   sorted A2AB   srtbad A30E
srtcal A2DA   titdun A0E5   title  A3FE   titlop A0C7   tmpfil A6BC
warms  CD03
```

```
   File name    Size    Disks    Sample INDEX Output   27 Oct 81

 PORT     .BAK     2   34
 PSP      .BAK    34   38
 UPCASE   .BAK     6    3
 PATCH    .BIN     3   35
 SORTSPEC .BIN     2    3
 ASMB     .CMD    47    1    4   12   33
 CHECK    .CMD     1   37
 CLOCK    .CMD     5   35   37
 CLOCKSET .CMD     8   37
 CLOK91FA .CMD     8   35
 COPY     .CMD     5    1    3    4   12
 COPYNEW  .CMD     6   37
 COPYB    .CMD     1    3
 DELETE   .CMD     2    1    3   12   33
 DIR      .CMD     5    1    3   12   33   37
 DUMP     .CMD     1   37
 EDIT     .CMD    20    1    4   33   37
 FILES    .CMD     3   37
 FILETEST .CMD     5    3
 FIND     .CMD     1    1   37
 FINDHI   .CMD     1    1   35
 FIX      .CMD     7    1
 FLAW     .CMD     4    3
 FREE     .CMD     1   37
 INDEX    .CMD     6    3
 LINK     .CMD     1    1    3   35
 LIST     .CMD     3    1    3   12   33   35
 MAP      .CMD     1    1   37
 MEMEND   .CMD     1   37
 MEMTEST  .CMD     4    3
 MODEM    .CMD     4   37
 MODGET   .CMD     3    9
 N        .CMD     1    1
 NEWDISK  .CMD     7    1    8   33   35
 O        .CMD     3    1   34
 OUT      .CMD     2   37
 PCOPY    .CMD     5    1   37
 PDEL     .CMD     3    1   37
 PORT     .CMD     1    1   38
 PR       .CMD    23    1    8   12   33
 PRINT    .CMD     2   37
 PSORT    .CMD     3    3
 PSP      .CMD     5    1   38
 QCHECK   .CMD     4    1   38
 QUICK    .CMD     3    3
 RN       .CMD     2    1
 RUN      .CMD     2   37
 S        .CMD     3    1    3    4   12   33   35   37
 SET      .CMD     1   37
 SPLIT    .CMD     1   37
 TEST     .CMD     5    3
 TTYSET   .CMD     2    1    3
```

```
   File name    Size    Disks    Sample INDEX Output   27 Oct 81

 UPCASE   .CMD     1    3
 Y        .CMD     1    1
 FLEX     .SYS    27    1    8   12   35   37
 PRINT    .SYS     1    1   37
 SRTMRG   .SYS    19    3
 CLOCK    .TXT    40   35
 CLOK91FA .TXT    78   35
 COMPUWAR .TXT     5    1
 FIND     .TXT     3    1
 FINDHI   .TXT     3    1
 INDEX    .TXT    33    3
 MODGET   .TXT     9    3
 O        .TXT     9   34
 PATCH    .TXT    25   35
 PORT     .TXT     2   38
 PSP      .TXT    30   34
 STARTUP  .TXT     1    1    4   12   35   37
 TEMP     .TXT    12    1
 UPCASE   .TXT     6    3
 WILL     .TXT    23    1
```

LOG

The 'LOG' utility

By: Nico C. Yssel
Elger 20,
1141 CD Monnickendam
Netherlands
Tel: 02995-4208

The programs, which together form the LOG utility, will enable the FLEX user to copy all terminal I/O to a file.

It is often useful, especially in tricky situations, to be able to see what you have been doing. Therefore I decided to write a LOG command for FLEX. The version of FLEX I am currently using is 2.8:3, the programs are written in standard TSC 6809 assembler. As LOG.CMD relocates itself, the programs cannot be used on a 6800 FLEX version.

The syntax to invoke the log function is:

LOG <fd>

in which <fd> is a standard FLEX file descriptor. The extension defaults to '.LOG'. Do not use an extension, as it will be overwritten anyway.

To end the log function enter:

LOG OFF

it is obvious that the only log file name you cannot use is 'OFF.LOG'.

The program 'LOG.CMD' will open a log file, it then relocates itself below MEMEND, setting MEMEND to the new value, it will save and overlay the vectors INCH2, OUTCH2 and FMSCLS, then it returns to FLEX. If the file descriptor happens to be 'OFF', the reverse takes place, and the log file will be closed. The ovelay of the vector INCH2 was necessary in my system, as the echo of input characters is done by the monitor, bypassing the OUTCH2 trap. In other systems the echo may be done in a diffenrent way, using the OUTCH2 vector, in which case the input part of the program can be removed (see the comments in the source listing).

To notify the user of the fact that a log file is running, the FLEX prompt will change to '>>>' as long as the log file is active.

As the log function tends to take up a lot of disk space, I have included a command XL.CMD, which removes files with the extension '.LOG', approximately in the same way as the command 'XOUT'. I have included XL into my STARTUP file, thus automaticcally removing all log files at system startup time.

A LOGLIST.CMD has been included, using .LOG as default extension and suppressing the extra line feed of LIST.CMD.

Even if you feel that you have no use for a log function, it is fun to try it out.

N.B. Some progr s do not use GETCHR or PUTCHR !!

```
*****************  *************************
*                                           *
* LOG COMMAND (LOG.CMD) WITH AUTOMATIC      *
* LOGGING OF THE CONSOLE DIALOGUE.          *
*                                           *
* THIS PROGRAM NEEDS THE COMMAND XL TO DELETE *
* OLD LOG FILES.                            *
*                                           *
* USE XL IN YOUR STARTUP FILE               *
* USE LOGLIST TO LIST THE LOG FILE          *
*                                           *
*        BY Nico C. Ijssel.                 *
* ELGER 20, 1141 CD MONNICKENDAM,           *
* NETHERLANDS                               *
* TEL 02995-4208                            *
```

```
*                                           *
*!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!*
* N.B. N.B. N.B. N.B. N.B. N.B. N.B. N.B. N.B. *
*!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!*
* CODE, NECESSARY TO CHANGE 'INCH2' IS      *
* PRECEDED BY IF MECHO AND FOLLOWED BY ENDIF *
* SET MECHO EQU 0 IF CODE IS NOT REQUIRED    *
*!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!*
*                                           *
0001 MECHO   EQU   1        MONITOR ECHO FLAG ON
*                                           *
*********************************************
*
* FLEX LABEL EQUATES

CD03 WARMS   EQU   $CD03
CD15 GETCHR  EQU   $CD15
CD1E PSTRNG  EQU   $CD1E
CD20 GETFIL  EQU   $CD20
CD3F RPTERR  EQU   $CD3F
D406 FMS     EQU   $D406
D409 FCBBAS  EQU   $D409
CC2B MEMEND  EQU   $CC2B

* ASCII CODE EQUATES

0004 EOT     EQU   $04
0004 FNAM    EQU   4
000C EXT     EQU   12
000D CR      EQU   $0D
0020 SPACE   EQU   $20

* EXTERNAL LABEL EQUATES

CC00 SYSFLG  EQU   $CC00
CC43 RETADR  EQU   $CC43
CD13 OUTCV   EQU   $CD13    OUTCH2 WILL BE CHANGED !!!!!!
*
*
       CD00 INCHCV  EQU   $CD00    INCH2 WILL BE CHANGED !!!!!!
*
*
       D404 CLSVEC  EQU   $D404
       CCAC PROMPT  EQU   $CCAC


C100                ORG   $C100


C100 20  05     START0  BRA   START1   BRANCH OVER VERSION
C102 81 2E 30 3A        FCB   $81.$2E,$30.$3A,$80 V = 1.0:0
C106 80
C107 86  01     START1  LDA   #1
C109 B7  CC00           STA   SYSFLG   SET NON-ZERO
*
*********************************************
*                                           *
* THE FOLLOWING CODE GETS THE LOG FILE      *
* NAME AND PUT IT INTO THE FCB.             *
* IF  THE LOG FILE NAME IS 'OFF' THEN       *
* THE LOGGING ACTION WILL BE TERMINATED     *
*                                           *
*********************************************
*
C10C 30  8D 0190        LEAX  LOGFCB,PCR
C110 BD  CD20           JSR   GETFIL   GET THE FILE NAME
C113 E6  04             LDB   FNAM,X
*
*********************************************
*                                           *
* THE NEXT CODE CHECKS TO SEE IF THE FILE   *
* NAME IS 'OFF', FOLLOWED BY ALL ZERO'S     *
*                                           *
*********************************************
*
C115 1083 6F46          CMPD  #('O'*256+'F')
C119 26  10             BNE   LOGST
C11B EC  06             LDD   FNAM+2,X
C11D 1083 4600          CMPD  #('F'*256)
C121 26  08             BNE   LOGST
C123 EC  08             LDD   FNAM+4,X
```

```
C125 E3   0A            ADDD   FNAM+6,X
C127 1027 008F          LBEQ   LOGEND
          C12B LOGST    EQU    *
C12B CC   4C46          LDD    #(256*'L+'O)
C12E ED   0C            STD    EXT,X
C130 86   47            LDA    #'G
C132 ED   0E            STD    EXT+2,X
                        *
                        ****************************************
                        *                                      *
                        * SAVE THE SYSTEM OPTIONS AND POINTERS,  *
                        * TO  BE RESTORED AFTER LOG HAS ENDED.   *
                        *                                      *
                        ****************************************
                        *
C134 FC   CC43          LDD    RETADR
C137 ED   8C 0128       STD    L_RETA,PCR
C13B FC   CD13          LDD    OUT2V     MUST BE REDIRECTED
C13E ED   8D 0123       STD    L_OVEC,PCR
                        *
                        *
C142 FC   CD00          LDD    INCH2V    MUST BE REDIRECTED FOR ECHO TO FILE
C145 ED   8D 011E       STD    L_IVEC,PCR
                        *
                        *
C149 FC   D404          LDD    CLSVEC    AS ABOVE
C14C ED   8D 0119       STD    L_CLSV,PCR
                        *
                        ****************************************
                        *                                      *
                        * NOW  GET THE MEMORY END, SAVE IT, AND  *
                        * SUBTRACT THE SIZE OF THE RELOCATABLE   *
                        * PART TO MAKE SPACE FOR LOG AND THEN    *
                        * RELOCATE IT.                           *
                        *                                      *
                        ****************************************
                        *
C150 FC   CC2B          LDD    MEMEND    GET END OF MEMORY
C153 ED   8D 010A       STD    L_MEND,PCR SAVE IT
C157 83   01E7          SUBD   #PISIZE   GET SIZE OF PART 2
C15A 25   3F            BCS    MZLOW     IF CARRY SET PISIZE > MEMORY
C15C FD   CC2B          STD    MEMEND    CAN DO
C15F 1F   02            TFR    D,Y       NEW MEMEND => Y
C161 31   21            LEAY   1,Y       START IS 1 HIGHER
C163 DC   01E7          LDD    #PISIZE   GET SIZE AGAIN
C166 34   22            PSHS   A,Y       SAVE M.O. SIZE AND XFER ADDRESS
C168 30   8D 009A       LEAX   PIPART,PCR X POINTS TO PART 2
C16C 50                 TSTB             < $100 TO MOVE?
C16D 27   03            BEQ    DECMSB    YES

X16F 17   0011    MOVLSB LBSR   MOVEXY    MOVE $100 BYTES

C172 6A   E4      DECMSB DEC    0,S       DEC M.O. SIZE
C174 2A   F9             BPL    MOVLSB    SOME LEFT?
C176 CC   3E3E           LDD    #$3E3E    LOG PROMPT IS >>>
C179 FD   CCAC           STD    PROMPT
C17C B7   CCAE           STA    PROMPT+2
C17F 35   82             PULS   A,PC      FORMER Y NOW IN PC
                        *
C181 C6   0C      MOVE12 LDB    #12
                        *
C183 A6   80      MOVEXY LDA    0,X+
C185 A7   A0             STA    0,Y+
C187 5A                  DECB
C188 26   F9             BNE    MOVEXY
C18A 39                  RTS

C18B 30   8D 0009 MZLOW  LEAX   MSGA,PCR

C18F 8D   C01E    ERROUT JSR    PSTRNG
C192 31   8D 00C7        LEAY   OUTPUT,PCR
C196 20   39             BRA    RESTOR

C198 2D 2D 20 4E  MSGA   FCC    '-- Not enough memory for a log --',EOT
C19C 6F 74 20 65
C1A0 6E 6F 75 67
C1A4 68 20 6D 65
C1A8 6D 6F 72 79
C1AC 20 66 6F 72
C1B0 20 61 20 6C
C1B4 6F 67 20 20
C1B8 2D 04
                        *
          C1BA LOGEND   EQU    *
                        *
C1BA 108E CD13          LDY    OUT2V     GET ADDRESS OF LOG OUTPUT
C1BE AE   A9 000E       LDX    O_FCB,Y   GET THE FCB ADDRESS
C1C2 86   04            LDA    #4
C1C4 A7   84            STA    0,X       INSERT CLOSE FILE CMD
C1C6 BD   D406          JSR    FMS
C1C9 27   06            BEQ    RESTOR
C1CB BD   CD3F          JSR    RPTERR
C1CE 7E   CD03          JMP    WARMS
                        *
          C1D1  RESTOR  EQU    *
C1D1 EC   A9 0002       LDD    JUMOFS,Y  FIND THE START OF LOG
C1D5 1083 CC2B          CMPD   #MEMEND
C1D9 26   03            BNE    NOMECH
C1DB FD   CC2B          STD    MEMEND

          C1DE  NOMECH  EQU    *
C1DE EC   A9 0006       LDD    O_RETA,Y
C1E2 FD   CC43          STD    RETADR
C1E5 EC   A9 0008       LDD    O_OVEC,Y
C1E9 FD   CD13          STD    OUT2V
                        *
                        *
C1EC EC   A9 000A       LDD    O_IVEC,Y
C1F0 FD   CD00          STD    INCH2V
                        *
                        *
C1F3 EC   A9 000C       LDD    O_CLSV,Y
C1F7 FD   D404          STD    CLSVEC
C1FA CC   2B2B          LDD    #$2B2B    ORIGINAL PROMPT IS +++
C1FD FD   CCAC          STD    PROMPT
C200 B7   CCAE          STA    PROMPT+2
                        *
C203 7E   CD03          JMP    WARMS
                        *
C205  LSTMEM  EQU   *-1
      *
C206 30   8C FD   PIPART LEAX  PIPART,PCR POINT TO OWN END
C209 30   1F             LEAX  -1,X
C20B AF   8D 0050        STX   OWNEND,PCR
C20F 30   8D 009A        LEAX  LOGFCB,PCR POINT TO FCB
C213 AF   8D 0054        STX   L_FCB,PCR SAVE IT
C217 86   02             LDA   #2        OPEN FOR WRITE
C219 A7   84             STA   0,X
C21B BD   D406           JSR   FMS       DO FLEX CALL
C21E 27   25             BEQ   LOGOK     OK?
C220 30   8D 0005        LEAX  MSGS,PCR
C224 8D   CD1E           JSR   PSTRNG
C227 20   A9             BRA   RESTOR

C229 2D 2D 20 43  MSGS   FCC   '-- Can't create an log file',EOT
C22D 61 6E 27 74
C231 20 63 72 65
C235 61 74 65 20
C239 61 6E 20 6C
C23D 6F 67 20 66
C241 69 6C 65 04
                        *
                        ****************************************
                        *                                      *
                        * NOW  OVERLAY THE VECTORS FOR PUTCHR    *
                        * AND  FOR FMSCLOSE WITH LOG ENTRIES.    *
                        *                                      *
                        ****************************************
                        *
C245 30   8D 0034 LOGOK  LEAX  OCLOSE,PCR
C249 BF   D404           STX   CLSVEC
C24C 30   8D 0000        LEAX  OUTPUT,PCR
C250 BF   CD13           STX   OUT2V
                        *
                        *
C253 30   8D 0046        LEAX  INPUT,PCR
C257 BF   CD00           STX   INCH2V
                        *
                        *
C25A 7E   CD03           JMP   WARMS
```

```
C25D 20    0E         OUTPUT  BRA    OUTPT1
                      *

C25F                  OWNEND  RMB    2
C261                  L_MEND  RMB    2
C263                  L_RETA  RMB    1
C264                  L_RETB  RMB    1
C265                  L_OVEC  RMB    2
                      *
                      *
C267                  L_IVEC  RMB    2
                      *
                      *
C269                  L_CLSV  RMB    2

C26B                  L_FCB   RMB    2
                      *
           0002       OWNOFS  EQU    OWNEND-OUTPUT
           0004       O_MEND  EQU    L_MEND-OUTPUT
           0006       O_RETA  EQU    L_RETA-OUTPUT
           0007       O_RETB  EQU    L_RETB-OUTPUT
           0008       O_OVEC  EQU    L_OVEC-OUTPUT
                      *
                      *
           000A       O_IVEC  EQU    L_IVEC-OUTPUT
                      *
                      *
           000C       O_CLSV  EQU    L_CLSV-OUTPUT
           000E       O_FCB   EQU    L_FCB-OUTPUT
                      *
C26D 34    16         OUTPT1  PSHS   A,B,X
C26F 30    8D 003A    PRCHAR  LEAX   LOGFCB,PCR
C273 BD    D406               JSR    FMS         OUTPUT TO FILE
C276 30    8C EC              LEAX   L_OVEC,PCR
C279 AD    94                 JSR    [0,X]

C27B 35    96         OUTRTN  PULS   A,B,X,PC
C27D 34    16         OCLOSE  PSHS   A,B,X
C27F BE    D409               LDX    FCBBAS      1ST LINK IN X
C282 27    F7                 BEQ    OUTRTN      NO OPEN FILES

C284 EC    84         CLLIST  LDD    0,X         GET ADDRESS OF NEXT LINK
C286 34    06                 PSHS   A,B         SAVE IT ON THE STACK
C288 30    88 E4              LEAX   -28,X       GET THE RELATED FCB
C28B AC    8C DD              CMPX   L_FCB,PCR   SAME AS WE USED?
C28E 27    07                 BEQ    IGNORE      YES, IGNORE
C290 86    04                 LDA    #4          CODE FOR CLOSE FILE
C292 A7    84                 STA    0,X
C294 BD    D406               JSR    FMS         DO CLOSE FUNCTION

C297 AE    E1         IGNORE  LDX    0,S++       GET POINTER BACK FROM THE STACK
C299 26    E9                 BNE    CLLIST      MORE?
                      *
C29B 35    96                 PULS   A,B,X,PC
                      *
                      *
                      *
C29D 34    14         INPUT   PSHS   B,X
C29F 30    8C C5              LEAX   L_IVEC,PCR
C2A2 AD    94                 JSR    [0,X]       DO NORMAL INPUT
C2A4 30    8D 0005            LEAX   LOGFCB,PCR
C2A8 BD    D406               JSR    FMS         OUTPUT ECHO TO FILE
C2AB 35    94                 PULS   B,X,PC
                      *
                      *
                      *
           C2AD       LOGFCB  EQU    *
C2AD                          RMB    $40         FCB DESCRIPTOR
           C2ED       OUTBUF  EQU    *

C2ED                          RMB    $100        FCB BUFFER
           01E7       PISIZE  EQU    *-PIPART
                      *

                              END    STARTO

1 ERROR(S) DETECTED

SYMBOL TABLE:

CLLIST C284   CLSVEC D404   CR     000D   DECMSB C172   EOT    0004
ERROUT C18F   EXT    000C   FCBBAS D409   FMS    D406   FMAN   0004
GETCHR CD15   GETFIL CD2D   IGNORE C297   INCH2V C000   INPUT  C29D
LOGEND C1BA   LOGFCB C2AD   LOGOK  C245   LOGST  C128   LSTMEM C205
L_CLSV C269   L_FCB  C26B   L_IVEC C267   L_MEND C261   L_OVEC C265
L_RETA C263   L_RETB C264   MCLOW  C18B   MECHO  0001   MEMEND CC2B
MOVE12 C181   MOVEXY C183   MOVLSB C16F   MSGA   C198   MSGS   C229
NOMECH C1DE   OCLOSE C27D   OUT2V  CD13   OUTBUF C2ED   OUTPT1 C26D
OUTPUT C25D   OUTRTN C27B   OWNEND C25F   OWNOFS 0002   O_CLSV 000C
O_FCB  000E   O_IVEC 000A   O_MEND 0004   O_OVEC 0008   O_RETA 0006
O_RETB 0007   PIPART C206   PISIZE 01E7   PRCHAR C26F   PROMPT CCAC
PSTRNG CD1E   RESTOR C1D1   RETADR CC43   RPTERR CD3F   SPACE  0020
STARTO C100   START1 C107   SYSFLG CC00   WARMS  CD03
```

```
                   ****************************************
                   *                                      *
                   * LOGLIST, A UTILITY TO LIST LOG FILES *
                   * BY:                                  *
                   * NICO C. VSSEL                        *
                   * ELGER 20                             *
                   * 1141 CD MONNICKENDAM                 *
                   * NETHERLANDS                          *
                   * TEL. 02995-4208                      *
                   *                                      *
                   ****************************************
                              OPT    PAG,NOG

           D403       FMSCLS  EQU    $D403
           D406       FMSCAL  EQU    $D406
           CD2D       GETFIL  EQU    $CD2D
           CD33       SETEXT  EQU    $CD33
           CD3F       RPTERR  EQU    $CD3F
           CD18       PUTCHR  EQU    $CD18
           CD03       WARMS   EQU    $CD03
           C840       FCB     EQU    $C840
           0008       EOF     EQU    $08
           000C       EXT     EQU    $0C

C100                          ORG    $C100
           C100       START   EQU    *
C100 20    01                 BRA    BEGIN
C102 12                       NOP
                      *
           C103       BEGIN   EQU    *
C103 8E    C840               LDX    #FCB
C106 BD    CD2D               JSR    GETFIL
C109 24    09                 BCC    FILOK
C10B 8E    C840               LDX    #FCB
C10E C6    15                 LDB    #$15
C110 E7    01         SAVERR  STB    $01,X
C112 20    27                 BRA    ERREPT
C114 CC    4C4F       FILOK   LDD    #(256*'L+'O) EXTENTION IS LOG
C117 ED    0C                 STD    EXT,X
C119 86    47                 LDA    #'G
C11B A7    0E                 STA    EXT+2,X
C11D 86    01                 LDA    #1
C11F A7    84                 STA    0,X
C121 BD    D406               JSR    FMSCAL      OPEN FOR READ
C124 25    15                 BCS    ERREPT
C126 4F                       CLRA               SET TO NEXT CHAR COMMAND
C127 A7    84                 STA    0,X         PUT IN FCB
C129 BD    D406       GETNXT  JSR    FMSCAL      GET 1 CHARACTER FROM THE FILE
C12C 27    08                 BEQ    PRNTIT
C12E E6    01                 LDB    1,X
C130 C1    08                 CMPB   #EOF
C132 26    DC                 BNE    SAVERR
C134 20    08                 BRA    RETURN
C136 BD    CD18       PRNTIT  JSR    PUTCHR
C139 20    EE                 BRA    GETNXT      LOOP
C13B BD    CD3F       ERREPT  JSR    RPTERR
C13E BD    D403       RETURN  JSR    FMSCLS

C141 7E    CD03               JMP    WARMS

                              END    START

0 ERROR(S) DETECTED
SYMBOL TABLE:

BEGIN  C103   EOF    0008   ERREPT C13B   EXT    000C   FCB    C840
FILOK  C114   FMSCAL D406   FMSCLS D403   GETFIL CD2D   GETNXT C129
PRNTIT C136   PUTCHR CD18   RETURN C13E   RPTERR CD3F   SETEXT CD33
SAVERR C110   START  C100   WARMS  CD03
```

```
                    *       ERASE LOG FILES AUTOMATICCALLY WITHOUT PROMPT
                            OPT   PAG,NOG
*** ILLEGAL OPTION

            C840  SYSFCB  EQU   $C840
            CC02  TTYEOL  EQU   $CC02
            CC11  LSTTRM  EQU   $CC11
            CD03  WARMS   EQU   $CD03
            CD1E  PSTRNG  EQU   $CD1E
            CD3F  RPTERR  EQU   $CD3F
            CD48  INDEC   EQU   $CD48
            D406  FMS     EQU   $D406
            CC0C  W_DRV   EQU   $CC0C
            0004  EOL     EQU   4

C100                      ORG   $C100

C100 20  05       START   BRA   START1
C102 81 2E 80 3A          FCB   $81,$2E,$80,$3A,$80
C106 80
C107 20  05       START1  BRA   START2
C109              WORK0   RMB   1
C10A              WORK1   RMB   1
C10B              L_DRV   RMB   1
C10C              LC106   RMB   1
C10D              LC107   RMB   1
          C10E    START2  EQU   *
C10E 8E  C1C9             LDX   #MESSG
C111 BD  CD1E             JSR   PSTRNG
C114 B6  CC11             LDA   LSTTRM
C117 81  00               CMPA  #$00
C119 27  14               BEQ   SET_W
C11B B1  CC02             CMPA  TTYEOL
C11E 27  0F               BEQ   SET_W
C120 BD  CD48             JSR   INDEC
C123 25  51               BCS   BAD_D
C125 9F  C109             STX   WORK0
C128 B6  C10A             LDA   WORK1
C12B B4  03               ANDA  #3
C12D 20  03               BRA   ST_DRV
C12F B6  CC0C     SET_W   LDA   W_DRV
C132 B7  C10B     ST_DRV  STA   L_DRV
C135 8E  8040             LDX   #SYSFCB
C138 A7  03               STA   3,X
C13A 9E  C840     G_SYS   LDX   #SYSFCB
C13D 86  06               LDA   #6        OPEN DIRECTORY
C13F A7  84               STA   0,X
C141 BD  D406             JSR   FMS
C144 26  2A               BNE   ERROR
XC146 BD  C187    INFLP   JSR   GTINF
C149 26  34               BNE   TSTEOF
C14B 6D  04               TST   4,X
C14D 2B  F7               BMI   INFLP     DELETED FILE
C14F 27  22               BEQ   LEAVE     END OF DIRECTORY
C151 EC  0C               LDD   12,X
C153 1083 4C4F            CMPD  #(256*'L+'O)
C157 26  ED               BNE   INFLP
C159 A6  0E               LDA   14,X
C15B B1  47               CMPA  #'G
C15D 26  E7               BNE   INFLP
XC15F BD  C191            JSR   TSPOOL
C162 27  E2               BEQ   INFLP
C164 8E  C840             LDX   #SYSFCB
C167 86  0C               LDA   #$0C      DELETE CODE
C169 A7  84               STA   0,X
C16B BD  D406             JSR   FMS
C16E 27  CA               BEQ   G_SYS
C170 BD  CD3F     ERROR   JSR   RPTERR
C173 7E  CD03     LEAVE   JMP   WARMS
C176 8E  C840     BAD_D   LDX   #SYSFCB
C179 C6  1A               LDB   #$1A
C17B E7  01               STB   1,X
C17D 20  F1               BRA   ERROR
C17F A6  01       TSTEOF  LDA   1,X
C181 81  08               CMPA  #8        BOF?
C183 26  EB               BNE   ERROR
C185 20  EC               BRA   LEAVE
C187 8E  C840     GTINF   LDX   #SYSFCB
C18A 86  07               LDA   #7        GET INFO REC
C18C A7  84               STA   0,X
C18E 7E  D406             JMP   FMS

C191 86  C718     TSPOOL  LDA   #$C718
C194 B7  C10C             STA   LC106
C197 27  20               BEQ   LC1E3
C199 A6  03               LDA   3,X
C19B B7  C10D             STA   LC107
C19E EC  88 11            LDD   17,X
C1A1 BE  C719             LDX   #$C719
C1A4 10A3 01      LC1B8   CMPD  1,X
C1A7 26  0C               BNE   LC1CC
C1A9 34  02               PSHS  A
C1AB B6  C10D             LDA   LC107
C1AE A1  84               CMPA  0,X
C1B0 35  02               PULS  A
C1B2 26  01               BNE   LC1CC
C1B4 39                   RTS
C1B5 7A  C10C     LC1CC   DEC   LC106
C1B8 27  0C               BEQ   LC1E3
C1BA 30  04               LEAX  4,X
C1BC 8C  C840             CMPX  #SYSFCB
C1BF 26  03               BNE   LC1E1
C1C1 8E  C810             LDX   #$C810
C1C4 20  0E       LC1E1   BRA   LC1B8
C1C6 B6  01       LC1E3   LDA   #1
C1C8 39                   RTS
                  *
          C1C9    MESSG   EQU   *
C1C9 0D 0A 0A             FCB   $0D,$0A,$0A
C1CC 2A 2A 2A 2A          FCC   '**** REMOVAL OF OLD LOG FILES IN PROGRESS'
C1D0 20 52 45 4D
C1D4 4F 56 41 4C
C1D8 20 4F 46 20
C1DC 4F 4C 44 20
C1E0 4C 4F 47 20
C1E4 46 49 4C 45
C1E8 53 20 49 4E
C1EC 20 50 52 4F
C1F0 47 52 45 53
C1F4 53
C1F5 00 0A 04             FCB   $0D,$0A,EOL

                          END   START

1 ERROR(S) DETECTED

SYMBOL TABLE:

BAD_D  C176  EOL    0004  ERROR  C170  FMS    D406  GTINF  C187
INDEC  CD48  INFLP  C146  LC106  C10C  LC107  C10D  LC1B8  C1A4
LC1CC  C1B5  LC1E1  C1C4  LC1E3  C1C6  LEAVE  C173  LSTTRM CC11
L_DRV  C10B  MESSG  C1C9  G_SYS  C13A  PSTRNG CD1E  RPTERR CD3F
SET_W  C12F  START  C100  START1 C107  START2 C10E  ST_DRV C132
SYSFCB C840  TSPOOL C191  TSTEOF C17F  TTYEOL CC02  WARMS  CD03
WORK0  C109  WORK1  C10A  W_DRV  CC0C
```

GENERAL PURPOSE INTERFACE BUS

BY: J.C.MOORE
1 THE SPINNEY,
FLEET,
HANTS, ENGLAND

THE MC68488 AND THE GPIB

Introduction

When Commodore introduced the PET using the GPIB (General Purpose Interface Bus) as its principle interface and only means of connecting a printer or disk drives they also started a new departure in low-cost control and logging systems for laboratory instruments. Since then many other manufacturers have offered the same interface on their computer systems. I first became involved with the GPIB when asked to provide a way of transferring a large

Update to ELEKTRA!

Package #1	Package #2	Package #3
2 MHz 6809 CPU Board Super Floppy Controller OS-9™ w/Edit, Asm, Debugger	2 MHz 6809 CPU Board Super Floppy Controller 4K Humbug™ and Star-Dos™	2 MHz 6809 CPU Board Super Floppy Controller (No software)
$695.00	**$675.00**	**$550.00**

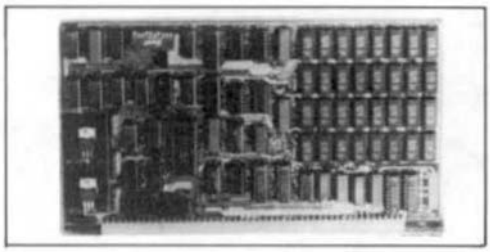
ELEKTRA OS-9™ with Editor, Assembler, and Debugger **$250.00**
ELEKTRA STAR-DOS™ (Adaptation Guide: $50.00) **$75.00**
OS-9™ Super Modem Program by Epstein Associates **$100.00**

ELEKTRA Super Floppy Controller
(Supports SD, DD, SS, DS, 5", 8", 1MHz, 2MHz, 8 Drives
Emulates the DC-1, DC-2, DC-3, #28, #38, #48, #58 Controllers
Perfect upgrade for the DC-4 **$295.00**
Drivers for TSC's versions of FLEX™ or STAR-DOS™ (user installed) **$30.00**
OS-9™ Drivers (Reads and writes CoCo format too, user installed) **$50.00**

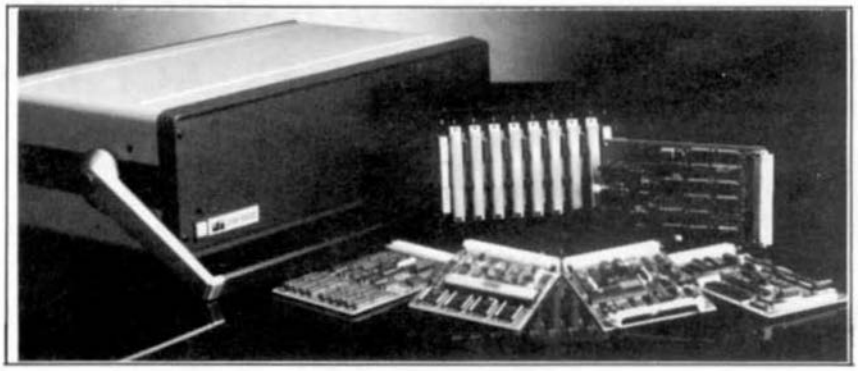
8" Floppy Drive Special w/manual, 90 day warranty
Siemens FDD 100-8 (SSDD) **$135.00**
Siemens FDD 200-8 (DSDD) **$185.00**

Removable Cartridge Winchester Drive (See next page for systems) **$1995.00**

2MHz Memory Boards with on board DAT
by Computer Excellence, Inc.
256K **$749.00** 512K **$1495.00**
1M **$2495.00**



Mizar 68000 — VME Development System with 256K RAM, 360K Floppy,
10 Mbyte Winchester, 4 Serial Ports (synchronous and/or asynchronous),
OS-9, Screen Editor, Assembler, "C" compiler, and SASI interface **$6495.00**



Phone:
AAA Chicago Computer Center
Technical Consultation available most weekdays from 4 p.m. to 6 p.m. CST
(312) 459-0450 120 Chestnut Lane Wheeling, IL 60090
See our catalog and ordering information on the next page.

ELEKTRA COMPUTER SYSTEM Includes chassis, dual port serial interface with two cables, CPU 8/9, 4K Humbug, 56K static RAM, super floppy controller with inboard ribbon cable, Star-Dos, dual 80 track DSDD floppy drives (other combinations available; phone). OS-9 may be substituted for HUMBUG and STAR-DOS. $2795.00

ELEKTRA COMPU ER CABINET THE LARGEST SS-50 COMPUTER CABINET AVAILAB E! Made of heavyweight 0.090" thick aluminum. Interior is 18-1/2" wide by 21-7/8" deep by 6-3/4" high. Heavy duty A.C. line cord. A.C. fuse holder. EMI filter. Fan with filter. Back panel has 10 cutouts for 'D' type data connectors. Front panel has on/off power switch, 2 illuminated push button switche (Reset and NMI/Abort), and two cutouts for 2 full height or 4 half height 5-1/4" disk drives. $250.00

RACKMOU T ELEKTRA COMPU ER CABI ET 17"w x 21.5"d x 6.7"h. Holds one full or two half height 5-1/4" floppy drives. $250.00

Filler Plate for drive opening: $10.00 Fan Filter: $10.00

POWER SUPPLY Highest quality linear power supply CONSERVATIVELY rated at 15a @ 8v, 3a @ 16v, 3a @ -16v. Multi-tapped primary for line tuning. $200.00

DISK REGULATOR BOARD WITH CABLES Standard version for 2 floppy drives $50.00
Heavy duty version for 1 Winchester drive and 1 floppy drive or 4 half heights $75.00

AUXILIARY POWER SUPPLY to power second Winchester drive $125.00

ELEKTRA UNIVERSAL SS-50/SS-50C MOTHERBOARD Heavyweight 0.125" thick. 18" long by 9" wide. 11 memory (50 pin) slots. 8 I/O (30 pin) slots. Complete address decoding and selection, as well as extended address capability, for I/O slots. Choice of 4, 8, or 16 addresses per I/O slot. 1" spacing between all memory and I/O slots. On board baud rate generator with low and high ranges providing jumper selectable rates of 75 through 38,400 for each of the five baud rate lines. slow device circuitry permitting 1 Mhz 30 pin disk controllers to run with 2MHz 50 pin CPU boards.
Mo nting hardware $5.00 Barebo rd w/documentation $80.00
Assembled w/tin connectors $380.00 Assembled w/gold connectors $480.00

ELEKTRA CHASSIS Includes cabinet, 110v power supply, power supply cables, standard disk regulator board with power cables, motherboard with gold square pin connectors, assembled and tested. (Add $25.00 for heavy duty reg lator.) $950.00

ELEKTRA 2MHz CPU 8/9 Use either the 6802 or 6808 (to run 6800 software) or 6809. Has provision for up to 3 2716 Eproms, 1K scratchpad, and MC6840 triple timer. Run OS-9™, FLEX™, STAR-DOS™. Bareboard $50.00 Assembled $275.00
Optional baud rate generator pro iding baud ra from 110 through 38,400 baud in two user selectable ranges $25.00

ELEKTRA DPS DUAL PORT SERIAL CARD Fit the standard 30 pin SS-50 bus I/O slot. Can be configured for 4 or 16 addresses per port. RTS, CTS, DTR, DCD, IRQ, FIRQ/NMI, and baud rate can be appropriately implemented for each port.
Bareboard $25.00 Assembled $95.00
C ble with jack socket assemblies (two needed per board) Each $25.00

ELEKTRA DPP DUAL PORT PARALLEL CARD Fits the standard 30 pin SS-50 bus I/O slot. Can be configured for 4 or 16 addresses per I/O slot. The direction of the TTL buffers can be controlled by either on board jumper connectors or by a signal from the peripherals. The interrupt request line for each port may be individually jumpered to either the IRQ or FIRQ/NMI bus line.
Bareboard $25.00 Assembled $80.00
Cable with jack socket assemblies (two needed per board) Each $25.00

ELEKTRA 64K STATIC RAM/RO MEMORY BOARDS with gold connectors (tin available) Assembled and tested With 56K RAM $269.00 With 64K RAM $299.00

ELEKTRA UNIVERSAL SUPER FLOPPY CONTROLLER HE BEST 30 PIN FLOPPY DISK CONTROLLER THAT YOU CAN BUY! Controls up to four 5-1/4" dr ves and four 8" drives for a total of eight system drives. Single density or double density, 1MHz or 2MHz 6800 or 6809. (Double density 8" requires 2MHz). Analog phase locked loop data separators with separate adjustments for 5" and 8" drives. Analog write precompensation circuit with separate adjustments for 5" and 8" drives. Designed to meet the data rate requirements of Western Digital floppy controller IC.
Bareboard (experts only) $100.00 Assembled and tested $295.00
Disk with drivers, setup, and formatting utilities. Specify FLEX 2.0, 6800 Gen. FLEX, FLEX 9.0, FLEX 9.1, 6809 Gen FLEX or STAR-DOS; 5" or 8" $30.00
Disk with drivers for OS-9 (Specify 5" or 8") $50.00

ELEKTRA WINCHESTER SYSTEMS THE BEST WINCHESTER SYSTEMS THAT YOU CAN BUY! Has automatic error detection and CORRECTION of up to 11 bit burst errors. SS-50 bus, extended addressing capabilities, DMA, on board sector buffer, drivers included for 6809 FLEX, STAR-DOS, or OS-9. Specify whose version of FLEX that you are using. Drivers for 6800 FLEX2 are available for an additional $100.00. Price includes host interface, controller, dirve(s), and cables.
7 Megabyte single drive sys. $1995.00 14 Megabyte dual drive sys. $2995.00
12 Megabyte single drive sys. $2295.00 24 Megabyte dual drive sys. $3595.00
19 Megabyte single drive sys. $2995.00 38 Megabyte dual drive sys. $4695.00
(19 Megabyte drives are the largest that can be supported by FLEX)
6 Megabyte removable cartridge single dr. sys. $2995.00 Drive only $1995.00
Circuit boards, cables, software (No drives) 995.00
SS-50C DMA B s Interface board only 695.00

ELEKTRA HD-5 Cabinet for dual 5 1/4" floppy drives with power supply, line cord, fuse power switch, and power cables to drives 150.00

ELEKTRA HD-5W As above but with EMI filter, fan, and heavy duty power supply. Powers 1 floppy and 1 Winchester or 4 half height 5" floppies 190.00

5' ribbon cable for dual outboard 5-1/4" disk rives 40.00
2' ribbon cable for dual inboard 5-1/4" disk drives 35.00
Custom cables available Phone

ELEKTRA HD-8 Dual 8" drive cabinet. EMI filter, fan with filter, power supply and power supply cables 350.00
Filler plate 10.00

6' ribbon cable for dual 8" disk drives 45.00

ELEK RA 30 PIN PROTOTYPING BOARD 20.00
ELEKTRA 50 PIN PROTOTYPING BOARD 40.00

GOLD 10 PIN CONNECTORS (Specify male with square pins or female) 1.50
TIN 10 PIN CONNECTORS (Specify male with square pins or female) .50

ELEKTRA is a trademark of AAA Chicago Computer Center.
FLEX and UniFLEX are trademarks of Technical System Consultants, Inc.
HELIX is a trademark of Hazelwood Computer Systems
HUMBUG, MICROBUG, and STAR-DOS are trademarks of STAR-KITS Software Systems Corp.
OS-9 and BASIC09 are trademarks of Motorola Inc. and Microware Systems Corp.

AAA CHICAGO COMPUTER CENTER (312) 459-0450
120 CHESTNUT LANE • WHEELING, IL 60090
Technical consultation available 4 PM to 6 PM most weekdays. Closed evenings and weekends.

TERMS Minimum order $20.00. Shipping and handling estimates within the Continental U.S., add 3% (MINIMUM $2.50). Illinois residents add 7% sales tax. We will refund your overestimated shipping and handling charges. Foreign shipping and handling, add 10% (MINIMUM $10.00). Foreign orders must be prepaid in U.S. dollars. Checks must be drawn on a U.S. bank. Heavy foreign items will be shipped air freight collect. Please phone between 4 PM and 6 PM weekdays if questions arise regarding shipping fees. Master Charge, Visa, and American Express honored.

Our apology. We are not staffed to answer technical inquiries through the mail. Please phone for technical help during the hours indicated above. The too frequent changing of our inventory and prices makes it uneconomical to publish a catalog. Our ads are intended to serve that purpose. Prices, specifications, and inventory are subject to change without advance notice.

SUPER MODEM PROGRAM Single character commands. No interrupts required. Transmit manually or transmit disk files (text) of any length to distant computer. Receive and save disk files (text) on local disk system. X-on/X-off supported. Tested for full duplex at speeds up to 9600 baud. Half duplex option. Echo option. Replaces CR with CR/LF (user option). Slow disk file transmit option.
Please specify 6800 or 6809, SSB, STAR-DOS™, or FLEX™, 5" or 8".
Instruction Man al and disk with both source and object code $75.00

OS-9 Super Modem Program by Epstein Associates with autodial, configuration file, etc. 100.00

ALL IN ONE
Editor — Text Processor — Mailing Labels — Mailing Lists — Multiple Form Letters
Use any CRT terminal and printer — Best Package For The Money Anywhere!
Specify 6800 or 6809, SSB, STAR-DOS™, or FLEX™, 5" or 8" 75.00
Add $35.00 for printed source listing, add $100 for source on disk.
All-In-One, Write'n spell, and Spell'n Fix package 250.00

Software by Technical Systems Consultants, Inc.	FLEX™ Source (List)	FLEX™ Source (Disk)	FLEX™ Man. Only	FLEX™ Object w/Man.	UniFLEX™ Add. Man.	UniFLEX™ Man. Only	UniFLEX™ Object w/Man.
Gen. FLEX w/Edit & ASMB	—	—	—	250	—	—	—
FLEX 9.1 (DC-2) w/Edit & ASMB	—	—	25	150	40	100	550
Advanced Programmers Guide	—	—	25	—	—	—	—
Editor	100	250	25	50	—	—	—
Assembler	150	250	25	50	—	—	—
Debug	175	250	25	75	—	—	—
Extended Basic	—	—	25	100	20	50	200
Basic Precompiler	—	—	25	50	10	25	150
Sort/Merge	—	—	25	75	20	35	150
Utilities	—	Inc.	25	75	10	25	150
Diagnostics	—	—	25	75	—	—	—
Text Processor	150	250	25	75	20	3	150
68000 X-ASMB on 6809	—	—	25	250	20	35	300
Pascal	—	—	50	200	25	50	300
Rel A MB/Linking L ader	—	—	25	150	20	35	175
6800 X-ASMB on 6809	—	—	—	100	—	—	—
Cobol	—	—	—	—	30	75	750
Fortran 77	—	—	—	—	35	65	450

Software by Microware Syste s Corp. (Suggested List Prices, varies/mfg)	Run-Time Pa kage	Source	Manual Only	Object w/Man.
OS-9™ Level 1 w/Edit, Asm, Debug		400.00	40.00	250.00
OS-9™ Level 2 w/Edit, Asm, Debug		400.00	40.00	500.00
OS-9™ Edit, Asm, Debug Pkg			25.00	125.00
Device Driver for Disk Controller (Specify Model)		100.00		
Device Driver for ACIA and PIA		50.00		
Clock Driver for 6840 and 58167 clock chips		35.00		
Entertainment Pack I, or File Handler Toolbox, or NineCom, or Virtual Disk Driver (Level 2 only)			10.00	85.00
Print Spooler (Level 2 only)			15.00	95.00
RMA Relocatable Macro Assembler			20.00	125.00
RMA/68000 Cross Assembler				400.00
BASIC09™ W/Run-Time	50.00	N/A	25.00	200.00
BASIC09 Tour Guide Book			18.95	
"C" Compiler			25.00	250.00
C Programming Language (Kernighan & Ritchie)			19.95	
CIS Cobol Compiler w/Forms 2 Prog. Gen.	50.00	N/A	40.00	400.00
Pascal Compiler	50.00	N/A	25.00	250.00
age Application Generator	300.00	N/A	25.00	995.00
Microware yearly support service (All products)				150.00
Edition UPdate wo/manuals 25.00			Version Update w/manuals	75.00

Special Software
STAR-DOS L1 (Specify ELEKTRA or DC 2, or DC-4) $75.00 Adaption guide $50.00
2K MICROBUG 40.00 4K HUMBUG 75.00 Custom versions $85.00
pell'n Fix by Peter Stark 178.58 Write'n Spell by Peter Stark 75.11
All-In-One, Spell'n Fix, and Write'n Spell package 250.00
SUPER SLEUTH Disassembler System ($101.00 for OS-9 version) 99.00

SD/DD DISK DRIVES 30 day guarantee	1 head Tandon	2 heads Tandon	2 heads CDC	1 head MPI	2 heads MPI
5-1/4", 40 tracks	225.00	300.00	300.00	250.00	325.00
5-1/4", 80 tracks	300.00	375.00	375.00	325.00	400.00

MPI or CDC Service Manual (pecify 40 or 80 track) 25.00 Qume DT-8 $50.00
Siemens 8" FDD 100-8 (S DD) $135.00 FDD 200-8 (DSDD) $165.00

SPECIAL BOARDS
Microtime II C lender and Clock Board (Assembled) 60.00
Data Mart 16K EPROM bareboard (2708 chips) 30.00

OUTBOARD EPROM PROGRAMMERS BY OPTIMAL TECHNOLOGY
Model EP-2A-79 (Personality modules extra) 169.00
Optimal Technology, Inc. 30 pin parallel I/O board for EP-2A-79 37.00
FLEX™ Software package for EP-2A-79 (Specify 6800 or 6809) 30.00
OS-9 Software package for EP-2A-79 10.00
Model EP-2B-87 (RS-232/20 MA, Motorola fmt, 8K buffer, 1200/9600 baud) 575.00
Model EP-2B-88-4 (Copies 1 to 4 EPROMS) 550.00
Personality/Copy Modules for 2708, 2716, 27C16, 2732, 27C32, 2732A, 2758, MCM68764, MCM68766, 2764, 27C64, 2764A, 27128, 27128A, 27256, 27C256, 2508, 2516, 2532, 256 , 25128, 2816, 2816A, 8702, 8751, 38E70 $17 to $39

Smoke Signal Broadcasting
DCB-4A Double Density Controller Board for 5" and 8" with DOS 549.00
SCB-69 6809 CPU Board 399.00

GIMIX CLEARANCE SALE	LIST	OUR PRICE		LIST	OUR PRICE
Cable (Par I/O)	24.95	20.00	6800 CPU board	224.03	100.00
80 X 24 Video Boards	398.76	250.00	Single prt ser, 1 cable	113.36	90.00
64 X 16 Video Boards	198.71	100.00	Dual prt par, 2 cables	138.32	110.00
16K Mem Bds w/cntrl reg.		145.00	Systems (trade-ins)		Phone
93L422 DAT chip	17.50	15.00			

HELIX
64K 6809 Computer $2395.00 Other computer systems available
DMA 5" and 8" Floppy Controller 495.00 6809 CPU Board 495.00
68008 board for SS-50 595.00 CP/M-68K 350.00

COMPUTED EXCELLE CE
2MHz Memory Board with on board DAT 256K $749.00 512K $1495.00 1M $2495.00

MIZAR
68000, VME development system with 256K RAM, 360K floppy, 10 Mbyte Winchester, 4 serial ports (synchronous and/or asynchronous), OS-9, screen editor, assembler, "C" co piler, and SA I interface 6495.00

SPECIALS
- SSB BFD Floppy Disk Controller (Version 3) Run FLEX or SSB DOS 100.00
- SWTPC DC-4* 230.00 MP-Mb (4K bareboard) 9.95
- SWTPC DMF-2* 595.00 S-32* RAM not included 124.50
- SWTPC MP-09 2MHz CPU* $295.00 *While supplies last
- High speed tape reader 50.00 300 Baud acoustic modem 129.00
- TI 810 rinter w/lower case and full vertical forms control 1200.00

WARNING AAA Chicago Computer Center does not provide repair or d agnostic service for customer assembled boards. AAA Chicago Computer Center does warranty and maintain service for our assembled boards.

number of BASIC programs from a PET to a Superbrain. I did this by fitting the Superbrain with a GPIB interface using the Motorola MC86488. File transfer was then fast and efficient.

More recently I designed an S30 bus interface card for Windrush which is now being sold in Europe and America. This article is written to provide an introductory explanation and some background on the GPIB which may be of general interest to all readers.

Bus Hardware

The GPIB was originated about 10 years ago by engineers at Hewlett Packard. It connects together between 2 and 15 devices in a free network. Normally one device is a system controller and the rest are talkers and listeners. Each talker/listener is assigned a unique device number (address). This is often set on DIP switches in the device. Additional devices can be serviced by having more than one control interface in the host computer and also by allocating secondary addresses to each primary address.

The bus carries 8 parallel data lines which allow transfers at up to 500000 bytes/second if the devices are separated by no more than 1 metre each with an end-to-end length of 15 metres, or 250000 bytes/second with separations of 2 metres and 20 metres end-to-end. As you can see this is pretty fast. In practice transfer speeds are usually limited by the software in the instruments and the controller. A second set of 8 lines carries 3 handshaking and 5 control signals.

Electrically each of the 16 lines is driven by a 3 state driver or an open collector driver capable of sinking 48 mA (eg a standard TTL 7433 or 7438) so that up to 15 drivers can feed the same line in a wired-AND configuration. This means that all drivers must be off for the line to go high, and thus the slowest active device on the bus can control the rate of transfer by holding the line low until it is ready. Each line also carries a line receiver, usually a differential device to minimise the effects of any noise picked up.

Specification

The GPIB is completely specified in IEEE488-1978 with a minor amendment published in 1980. This is a very comprehensive document. Luckily you do not have to be familiar with it to be able to use the bus! There is also an equivalent European spec IEC-625-1 which is not much used.

Uses

The bus may be used to connect together networks of computers for transferring files as mentioned already; also to connect a controlling computer to floppy disk & hard disk drives, printers, digital plotters, and a wide variety of laboratory instruments including signal generators, voltmeters, frequency counters, A to D converters and the like. This lets you construct versatile Automatic Test Equipment (ATE) installations.

Bus Messages

Message bytes over the bus take one of two forms:
1. Command bytes are sent by the controller. These are identified by making one of the control lines (ATN i.e. attention) true at the same time. There are quite a few possible commands. The most common ones are concerned with telling a selected device to talk or listen, and telling devices to unlisten or untalk. All devices on the bus must read all command messages, regardless of what they were doing before, and respond if required.

2. Data bytes are sent by any device when the ATN line is false. Often they will be in ASCII. Message strings may be terminated by a carriage return/line-feed, and also by making another control line (EOI end or identify) true.

Thus the majority of transfer sequences are of the form shown in the example below:

Source	Byte(hex)	ATN	EOI	Remarks
controller	46	true	-	Tell device 6 to be a talker
controller	27	true	-	Tell device 7 to be a listener
-	-	false	false	Controller releases bus
device 6	48	"	"	'H
"	45	"	"	'E
"	4C	"	"	'L
"	4C	"	"	'L
"	4F	"	"	'O
"	0D	"	"	C/R
"	0A	false	true	L/F + EOI
(controller recognises end of message and arranges an orderly close down:)				
controller	5F	true	false	Untalk all
controller	3F	true	false	Unlisten all

Throughout the above sequence the flow of data was controlled by the 3 handshake lines, which are: RFD ready for data, DAV data available, and DAC data accept. To complete the story there are 3 more control lines: SRQ service request, REN remote enable, and IFC interface clear. These 3 are not normally used in simple systems.

Implementation

The electrical interface to the GPIB may conveniently be provided by 2 Motorola MC3447 or 4 MC3448 bus transceiver ICs. These are a cost-effective way of meeting the full IEEE488 electrical spec. The control logic can be implemented in hardware or software. A software solution could use a program to drive two ports of a 6821 PIA. By this means all 16 lines can be toggled and read as necessary. This could be a cheap way of providing a partial subset of the GPIB capabilities. For a talker/listener it will not meet the IEEE spec of responding to the ATN line in 200nS, although this may not be important.

The simple hardware solution is to use the MC68488 which is a 40 pin IC programmed for the job. This will run at the full bus speed and handle all handshaking automatically. On the processor side it will be limited by the host software, although DMA could be used for really fast applications. The 68488 will interface directly to a 6800/6809 bus. It has 3 register select lines so will require 8 bytes of address space.

The only catch with this IC is that it does not provide any controller functions. These have to be supplied by additional external logic.

Program control of the 68488 is very simple as the following examples show:

```
WARMS  EQU $CD03
GETCHR EQU $CD15
PUTCHR EQU $CD18
```

* MC68488 addresses assuming card is in slot 2:

```
REG0    EQU $E020
REG2    EQU $E022
REG3    EQU $E023
REG4    EQU $E024
REG7    EQU $E027

MYDEVN EQU 4 My device number

EOFCHR  EQU  $1A Control Z, often marks end of text
files
```

```
                    *
                    * To read a file from another device:
                    *
0000                        ORG    $0000     Or wherever
                    *
0000 86     80      RFILE   LDA    #%10000000
0002 B7     E023            STA    REG3      Reset
0005 4F                     CLRA
0006 B7     E023            STA    REG3      Release reset
0009 86     04              LDA    #MYDEVN
000B B7     E024            STA    REG4      Primary address
000E 4F                     CLRA
000F B7     E022            STA    REG2      Default address modes
                    *
                    * Now initialised.
                    * Assume remote controller will make us a listener.
                    *
>0012 BD    005F    READ    JSR    BYTIN
                    *
                    * Display character, or write to disk file if using Flex "O" utility.
                    *
0015 BD     CD18            JSR    PUTCHR
0018 81     1A              CMPA   #EOFCHAR
001A 27     04              BEQ    WS
001C C5     02              BITB   #[illegible]0010 END detected ?
001E 27     F2              BEQ    READ
0020 7E     CD03    WS      JMP    WARMS
                    *
                    * End.
                    *
                    * Now send keyboard or a disk file to another device.
                    *
                    *
0023 86     80      SFILE   LDA    #%10000000
0025 B7     E023            STA    REG3      Reset
0028 4F                     CLRA
0029 B7     E023            STA    REG3      Release reset
002C 86     04              LDA    #MYDEVN
002E B7     E024            STA    REG4      Primary address
0031 4F                     CLRA
0032 B7     E027            STA    REG7      Default address modes
                    *
                    * Now initialised as before.
                    * Assume remote controller will make us a talker.
                    *
0035 BD     CD15    SEND    JSR    GETCHR    from keyboard or disk
0038 81     1A              CMPA   #EOFCHAR
003A 27     05              BEQ    SNDEOF
>003C BD    0050            JSR    BYTOUT
003F 20     F4              BRA    SEND
0041 34     02      SNDEOF  PSHA
0043 86     20              LDA    #%00100000
0045 B7     E023            STA    REG3      Force EOI
0048 35     02              PULA
>004A BD    0050            JSR    BYTOUT
004D 7E     CD03            JMP    WARMS
                    *
                    *
0050 34     02      BYTOUT  PSHA
0052 B6     E020    L7      LDA    REG0
0055 84     40              ANDA   #%01000000 Previous byte gone ?
0057 27     F9              BEQ    L7
0059 35     02              PULA
005B B7     E027            STA    REG7
005E 39                     RTS
                    *
005F B6     E020    BYTIN   LDA    REG0
0062 85     01              BITA   #[illegible]1 Byte available ?
0064 27     F9              BEQ    BYTIN
0066 1F     8940            TAB              Save END STATUS
0069 B6     E027            LDA    REG7      & release DAC line
006C 39                     RTS
                    *
```

Other Solutions

Some other manufacturers of GPIB ICs that I know of are:

1. Intel 8291 and 8292 chip set. These also require 2 8293s for the bus interface, or 5 MC3448s and some TTL.
2. Texas TMS9914. This requires a 75160A and a 75162A for bus interface.
3. Fairchild 96LS488. This is a 48 pin device which includes the bus interface. It is designed for stand alone (non program controlled) instruments. It would need additional logic to provide a controller function. Fairchild also second-source the 68488.

Of all these ranges the Motorola combination is the cheapest.

Windrush Card

If you don't wish to brew your own the Windrush card is a complete ready to go solution. It is a compact S30 bus PCB with gold plated edge connectors and all ICs socketed. It uses the MC68488 with additional logic to provide a complete controller/talker/listener facility. It is supplied with a comprehensive manual which includes circuit diagrams, all instructions, and full software listings. Also included is a full reprint of the Kilobaud articles (Reference 4 below).

References

Much more information on the GPIB is contained in the following sources:

1. Windrush IEEE488 Controller Manual.
2. Motorola MC68488 Data Leaflet.
3. Motorola "Getting aboard the 488-1975 bus".
4. "Get your PET on the IEEE488 bus" by G.Yob, Kilobaud, Jul Aug & Sep 1980.
5. "The PET and the IEEE488 bus", by E.Fisher & C.W.Jensen, published by Osborne/McGraw-Hill (ref no 0-931988-31-4).

- - -

BIT BUCKET

Non-keyboard Data Entry to a 6800 microcomputer.

A.J. Hall,
Diagnostic Ultrasound Unit.
Queen Mother's Hospital.
Glasgow, Scotland.

INTRODUCTION:

Digitisers or Graph tablets convert graphical coordinate data from a diagram, map, chart recording or echo into digital form for analysis or processing by computer. Coordinate points, which must lie within the active area of the digitiser, are specified by touching them with a pen-like stylus. The Summagraphics "Bitpad One" (Trade Mk.) is an example of what is available commercially; it operates on the magnetostrictive principle with a current pulse being sent along a "sound" wire lying at right angles to a magnetostrictive wire mesh laid beneath the pad surface. The current causes the mesh dimensions to change and the resultant strain wave is detected by coils within the stylus. An inbuilt microprocessor calculates the coordinate position of the pen from the time taken for the wave to reach the stylus. It also senses whether the stylus tip microswitch is closed or not and sets a flag accordingly. Depending on the type of Bitpad the data is then output, either in 8 bit parallel or in RS232 serial format.

MODES OF OPERATION

The mode and rate of sampling data points can be predetermined by internal switch settings or be selected under software control; the latter is more flexible especially if the Bitpad is interfaced to a microcomputer. The RS232 interfaced Bitpad is controlled by sending an ASCII character to it; this selects the desired sampling rate of coordinate pairs and the operational mode. The modes are as follows:- Point Mode - on depressing the stylus and closing the tip microswitch the bitpad outputs an X,Y coordinate pair and a flag for stylus up/down. Stream Mode - the flag and coordinate pair are output continuously while the stylus is either in contact with the pad or close to its surface. Switch Stream Mode - depression of the stylus to close the tip microswitch causes a stream of coordinate pairs and flags to be output until the stylus is lifted to open the tip microswitch even though the tip still remains in contact with the pad.

Data Format - RS232 serial and arranged as below

XXXX,YYYY,F CR LF

The line feed (LF) is optional and switch selectable; the data is in ASCII BCD format where X=X axis value Y=Y axis value, F=0 or 1 and flags the state of the tip microswitch.

INTERFACING THE BITPAD

I have used an RS232 Bitpad connected to a SWTP 6800 microcomputer via a serial interface at Port 0. The baud rate and stop bits of the Bitpad should, of course, correspond to that of the interface. Commas are used to separate each block of coordinate and flag data and the string is terminated with a carriage return. The format is identical to the keyboard input expected in response to a Basic "Input" statement so to communicate with the Bitpad only the I/O vector is

SWTBUG has to be changed - from Port 1 to Port 0. This is done by a "pOKE" statement to alter the contents of $A00B (40971 decimal) from 04 to 00. This alters the port address from $8004 to $8000.

Directly accessing the Bitpad from your Basic Program is this manner means that only the point mode can be used; it generates one set of coordinate data when the stylus is depressed, the other modes produce strings of data and these will not be acceptable. No machine code subroutines are necessary utilising SWTBUG INEEE input routine and detecting the carriage return to seperate the coordinate pairs and flags. However for many purposes the point mode is all that is needed e.g. selecting options on a menu or specifying specific points on a curve.

At the end of any access to Port 0 the I/O vector should be reset to Port 1 as the keyboard can be used for debugging the Program during development. The routines are as follows:

SELECT POINT MODE

This is done once at the start of your program unless you intend to change modes while the Program is running, in which case it should be called a subroutine.

```
POKE (40971,0)  REM SELECT PORT 0 I/O
PRINT "P"       REM P sets pad to point mode
POKE (40971,4)  REM I/O to Port 1
```

As we are working in Basic the port addresses have to be in decimal unless your Basic allows hex characters.

INPUT OF DATA FROM BITPAD

Having selected Point mode a General routine is needed to get data from the pad. One is given below.

```
POKE (40971,0)  REM I/O to Port 0.
INPUT X,Y,P     REM wait for input
pOKE (40971,4)  REM I/O back to Port 1.
```

These a few simple Basic commands allow the Bitpad to be accessed and data obtained for a program to operate on.

DATA ENTRY USING THE BITPAD

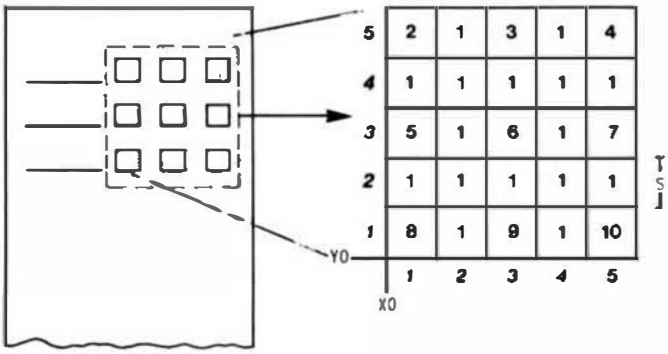
One example is the analysis of multiple choice questionnaires where the answers are indicated by ticks in the appropriate boxes. The form is placed against reference marks on the Bitpad so that the locations of the answer boxes are defined. One approach, which ensures that your Program will always respond in a predictable manner, is to divide the area or screen within which the answer boxes lie into the elements of a basic two dimensional array which I will call P(Y,X); see fig. 1. On entry to the program the array is filled with 1's and then specific locations are set with numbers in ascending order (2,3----etc,). Each number corresponds to a specific answer and so can control program branching using the Basic instruction on X GOTO----. If a blank area is accidentally touched P(Y,X) gives a 1 and the first address in the On X GOTO instruction is a routine which gives an error message and waits further input. The Bitpad has an active area of 11" x 11" and a resolution of .005", thus the coordinates can range from 0-2200 in both X and Y with the origin at the bottom lefthand corner. The values returned by the Bitpad will lie in this range and they must be scaled so that integer values of X and Y are produced to address the array p(Y,X). A typical example is given below.

```
X=X-XO          REM XO,YO are the answer area origins.
Y=Y-YO          REM X,Y are coordinates from stylus.
X=X/S           REM S is the length of one side of the
Y=Y/S           REM of the answer square is .005 units.

X=INT(X)+1      REM scale answer from 1 up
Y=INT(Y)+1
X=P(Y,X)
ON X GOTO-----
```

Additional traps can be inserted before the statement X=P(Y,X) to handle values greater or less than the array dimensions.

SUMMARY

The interfacing of a serial version of a graphics Bitpad via a MP-B interface located at Port 0 is simple and straightforward and it can be accessed directly from a Basic program by simple state-ments. The Bitpad will have to be set to suit the configuration of Port 0- some Basics set it to a configuration that is different to the SWTBUG configuration. If the Bitpad is used for data entry then a Basic array is a useful means of ensuring that your Progres behaves in a predictable manner if the stylus is placed to an area where there are no answer boxes. In addition a program can easily be altered to accommodate additions or deletions. Deletions require the appropriate value in the array to be replaced by a 1 while additions require the replacement of a 1 by the next free number in the ON X GOTO sequence and the addition of code to deal with the response. The examples given for generality have altered the I/O vector by means of POKE statements to access Port 0 but many Basics provide a "Port" statement to do this. However if you are using EXTP's Basic version 3.5 the following corrections are needed before the "POKE" statement can be used. Location $16dc should be altered from $26 to $20 and $1721 from $C8 to $CB.



Typical answer box on questionnaire with equivalent array P(Y,X,). XO, YO are co-ordinates of box origin: S = length of an answer box side in multiples of .005" units. Nos. 1-5 are normalised values used to address array P(Y,X). The value at each array position is shown in the middle of each box.

I have noticed that there has been very little published in the 68 MICRO JOURNAL relating to the UNIFLEX operating system. I purchased a copy for my computer in December 1983 and found it lacked a utility to modify the contents of a disk file. So, I am offering one for publication.

The program is an on screen editor which provides a dump of a 256 byte block of the file in both HEX and ASCII. To modify the file it is only necessary to move the cursor to the desired byte and then type in the desired modification in HEX. To allow for mistakes all modifications are placed into a buffer. When satisfied the changes can be saved by using the 'S' or 's' command. The program contains an extensive set of commands which I will not describe here as they are fully documented in the program. To obtain information about all commands it is only necessary to use the 'I' or 'i' command when the program is run.

Maximum use has been made of advanced VDT commands. This has the advantage of making an efficient program but has the disadvantage of making the program terminal dependent. For those who do not have an Datamedia Excel 42 the following lines will need to be modified.

```
1600 -- Position cursor at 'ar' - screen row
                           'ac' - screen column
5610 -- Erase from current position to end of line
6010 -- Video attribute on.
6510 -- Video attribute off
6610 -- Clear screen
```

I hope that this program will be of interest to your readers. Should any difficulties arise with this program I can be contacted at the address below and I would be pleased to hear from any other UNIFLEX user.

Yours Sincerely,

ALAN MILES
7 Anderson Street
Clifton Hill
Melbourne, 3068
Australia.


```
5 width 0
6 on error goto 9000
10 dim v(256),c$(21)
20 c$='0123456789ABCDEFabcdef'
25 fi$='' : fi$='' : rem file name
30 rem init HEX constant array
40 for x=0 to 21
50 c$(x)=mid$(c$,x+1,1)
60 next x
80 s=0
84 gosub 2000 : rem print header
86 gosub 3000 : rem process commands
90 if fi$='' goto 110
100 close 1
110 exit
1000 rem ------------------------------
1002 rem read & print a hex block
1004 rem entry
1006 rem        a - no. of byte at which to start reading
1008 rem exit
1010 rem        z - no. of lines read
1012 rem        c - no. of characters read in last line
1014 rem        b - no. of bytes read in block
1016 rem      v() - array of bytes read
1120 b=0
1130 print
1140 for z=1 to 16
1150 gosub 1370 : rem read upto 16 bytes of data from file
1160 if c=0 then z=z-1 : goto 1240
1170 gosub 1250 : rem display HEX
1180 gosub 1620 : rem display ASCII
1190 print
1200 b=b+c
1210 if c<>16 then goto 1240
1220 next z
1230 return
1240 for y=z+1 to 16
1242 sc=5 : sr=y+3 : gosub 1550
1244 gosub 5600 : rem erase eol
1246 next y
1248 return
1250 rem ------------------------------
1260 rem Print a line of HEX bytes
1270 rem c - number of bytes on line
1280 rem v(y+b) - current byte to convert
1290 rem y - byte counter
1300 rem ------------------------------
1310 sc=5 : sr=z+3 : gosub 1550
1320 for y=1+b to c+b
1330 q=int(v(y)/16) : v$=c$(q)+c$(v(y)-q*16)
1340 print ' ';v$;
1350 next y
1360 return
1370 rem ------------------------------
1380 rem read up to 16 bytes from file
1390 rem if less than 16 are available then c reflects no.
1400 rem ------------------------------
1405 if (a+b+10)>nb then 11=nb-a-b else 11=16
1410 for c=1 to 11
1420 v(c+b)=asc(f$(a+b+c-1))
1430 next c
1440 return
```

```
1450 rem ------------------------------
1460 rem convert DEC to HEX
1470 rem q - no. of 16's in byte
1480 rem v - no. to convert
1490 rem v$ - HEX answer
1500 rem ------------------------------
1510 q=int(v/16)
1520 v$=c$(q)+c$(v-q*16)
1530 return
1550 rem ------------------------------
1560 rem position cursor at specified screen position
1570 rem ac - screen column (1-80)
1580 rem ar - screen row (1-24)
1590 rem ------------------------------
1600 print chr$(27);"=";chr$(ar+31);chr$(ac+31);
1610 return
1620 rem ------------------------------
1630 rem display ASCII at end of line
1640 rem ------------------------------
1650 print tab(53);
1660 for x=b+1 to b+c
1670 if v(x)>31 and v(x)<127 then print chr$(v(x)); else print ".";
1680 next x
1682 if c=16 goto 1698
1684 for x=c+1 to 16
1686 print " ";
1688 next x
1690 return
2000 rem ------------------------------
2010 rem print header
2020 rem ------------------------------
2030 gosub 6600 : rem clear screen
2040 print "     0  1  2  3  4  5  6  7  8  9  A  B  C  D  E  F"
2050 print
2060 for x=0 to 15
2070 v=x : gosub 1450
2080 print v$;" - "
2090 next x
2100 print
2110 print "COMMAND";
2900 return
3000 rem ------------------------------
3002 rem enter commands & perform the appropriate action
3004 rem ------------------------------
3010 ar=21 : ac=9 : gosub 1550
3015 gosub 5600 : rem erase to end of line
3020 c$=inch$(0) : rem get a command
3030 if c$="l" or c$="L" goto 3100
3032 if c$="c" or c$="C" goto 3200
3034 if c$="q" or c$="Q" goto 3300
3035 if c$=chr$(8) goto 3400 : rem left shift
3036 if c$=chr$(31) goto 3410 : rem up shift
3037 if c$=chr$(12) goto 3420 : rem right shift
3038 if c$=chr$(22) goto 3430 : rem down shift
3039 if c$=chr$(30) goto 3440 : rem home
3040 if c$="n" or c$="N" goto 3500 : rem next block
3042 if c$="p" or c$="P" goto 3600 : rem previous block
3046 if c$="s" or c$="S" goto 3800 : rem save buffer & changes
3048 if c$="i" or c$="I" goto 3900 : rem list instructions
3050 goto 3700 : rem test for HEX & enter if yes
3097 e$="You have entered an invalid command. Type I for instructions." : gosub 5000
3099 goto 3000
3100 rem process file name
3105 if fi$<>"" then close 1
3110 input #0," ";fi$
3130 open old fi$ as 1
3140 position #1,0,mode 2, response mb : rem max. bytes in file
3150 close 1
3160 open fi$ as 1 size 1
3170 dim #1, f$(mb-1)=1
3180 s=0
3182 fi$=fi$ : rem save file name
3185 e$="The specified file has been loaded successfully." : gosub 5000
3200 rem display current hex block
3205 gosub 8500 : if f=0 then goto 3099
3210 gosub 1000
3212 c0=1 : c1=1 : gosub 7000 : rem display highlight byte
3215 e$="The current block has been displayed." : gosub 5000
3220 goto 3099
3300 rem quit
3310 gosub 6600 : rem clear screen
3320 return
3400 rem left shift
3402 c1=c0-1
3404 gosub 7000
3406 goto 3099
3410 rem up shift
3412 c1=c0-16
3414 goto 3404
3420 rem right shift
3422 c1=c0+1
3424 goto 3404
3430 rem down shift
3432 c1=c0+16
3434 goto 3404
3440 rem home
3442 c1=1
3444 goto 3404
3500 rem read next block
3510 if s+256 <= mb-1 then s=s+256 else e$="There is no next block, - BOTTOM OF FILE -" : gosub 5000 : goto 3099
3515 gosub 1000
3520 c0=1 : c1=1 : gosub 7000 : rem display highlight byte
3530 e$="Next block has been displayed." : gosub 5000
3540 goto 3099
3600 rem read previous block
3610 if s-256 >= 0 then s=s-256 else e$="There is no previous block, - TOP OF FILE -" : gosub 5000 : goto 3099
3620 gosub 1000
3630 c0=1 : c1=1 : gosub 7000 : rem display highlight byte
3640 e$="Previous block has been displayed." : gosub 5000
3650 goto 3099
3700 rem ------------------------------
3702 rem enter character check for HEX. if not return to
3704 rem command mode. Once two HEX digits entered convert to
3706 rem integer and save in buffer. Move to next buffer location.
3710 rem ------------------------------
3712 gosub 8500 : if f=0 goto 3099
3714 for x=0 to 21
3738 if c$=c$(x) goto 3746
3740 next x
3742 goto 3097
3746 rem ------------------------------
3748 h$=c$
3750 c$=inch$(0)
3752 for x=0 to 21
3754 if c$=c$(x) goto 3760
3756 next x
3758 goto 3742
3760 rem ------------------------------
3762 h$=h$+c$
3764 v(c0)=hex(h$)
3765 gosub 8000 : rem display entered HEX
3766 c1=c0+1
3768 gosub 7000 : rem shift highlight byte
3769 e$="Buffer has been modified and byte displayed." : gosub 5000
3770 goto 3099
3800 rem ------------------------------
3802 rem save current buffer onto file
3804 rem ------------------------------
3806 e$="Please WAIT the buffer is being saved." : gosub 5000
3808 gosub 8500 : if f=0 goto 3099
3810 for x=s to s+b-1
3820 f$(x)=chr$(v(x+1))
3830 next x
3840 e$="Current buffer has been saved into the file." : gosub 5000
3850 goto 3099
3900 rem ------------------------------
3902 rem print instructions
3904 rem ------------------------------
3910 gosub 6600 : rem clear screen
3914 print "COMMAND"
3916 print "-------"
3918 print
3920 print "C or C - Display current file block. If it is necessary to delete"
3922 print "        changes from buffer this command can be used. It will fill"
3924 print "        buffer from the file and overwrite changes."
3926 print "I or I - Display instructions and pointer information."
3928 print "L or L - Load and display the first block of a file."
3930 print "        eg. l nothing.b"
3932 print "N or N - Display the next block of the file."
3934 print "P or P - Display the previous block of a file."
3936 print "Q or Q - Quit program and return to operating system."
3938 print "S or S - Save current buffer and changes into the file."
3940 print "        This command must be used to modify any file."
3942 print
3944 print "        Arrow keys can be used to move cursor to any buffer position."
3946 print
3947 print "        Enter changes as HEX characters. Change occurs at cursor position."
3948 print
3950 print"File name - ";fi$,"Start of buffer in file - ";s+1
3952 print"Cursor position in file - ";s+c0,"Cursor position in buffer - ";c0
3954 print
3956 print tab(30);"Hit any key to continue."
3958 rem ------------------------------
3959 c$=inch$(0)
3960 gosub 2000 : rem print header
3962 gosub 3970 : rem display buffer
3963 goto 3899
3970 rem ------------------------------
3972 rem display current buffer
3974 rem ------------------------------
3975 if z=0 goto 3998
3976 z1=z : b1=b : c1=c : rem save counters
3978 b=0
3979 print
3980 for z=1 to z1
3982 if z=z1 then c=c1 else c=16
3984 gosub 1250 : rem display HEX
3986 gosub 1620 : rem display ASCII
3987 print
3988 b=b+c
3990 next z
3992 if z<>16 then gosub 1240
3994 c1=c0 : gosub 7000
3998 return
5000 rem ------------------------------
5002 rem print error message in e$
```

```
5004 rem ------------------------------
5010 gosub 5700 : rem erase status line & position cursor
5020 gosub 6000 : rem start flashing
5030 print e$;
5040 gosub 6500 : rem stop flashing
5050 return
5600 rem ------------------------------
5602 rem erase to end of line from current position
5604 rem ------------------------------
5610 print chr$(27);'T';
5620 return
5700 rem ------------------------------
5702 rem erase status line
5704 rem ------------------------------
5710 sr=23 : sc=15 : gosub 1550
5720 gosub 5600
5730 return
5000 rem ------------------------------
5002 rem print status message
5004 rem ------------------------------
5010 gosub 5700 : rem erase status line & position cursor
5020 print e$;
5030 return
6000 rem ------------------------------
6002 rem start flashing
6004 rem ------------------------------
6010 print chr$(27);'G4';
6020 return
6500 rem ------------------------------
6502 rem stop flashing
6504 rem ------------------------------
6510 print chr$(27);'G0';
6520 return

6600 rem ------------------------------
6602 rem clear screen
6604 rem ------------------------------
6610 print chr$(26)
6620 return
7000 rem ------------------------------
7010 rem shift highlight byte
7015 rem c1 - new position of highlight
7020 rem c0 - current position of highlight
7022 rem ------------------------------
7034 if c1>b then print chr$(7) : return
7036 if c1<1 then print chr$(7) : return
7040 gosub 7400 : rem remove highlight at c0
7045 c0=c1
7050 gosub 7500 : rem set       "       " c0
7060 return
7400 rem remove highlight at c0
7410 tm=int(c0/16.0001)
7420 sr=4+tm : sc=2+3*(c0-tm*16) : gosub 1550
7430 gosub 6500
7440 return
7500 rem set highlight at c0
7510 tm=int(c0/16.0001)
7514 sr=4+tm : sc=5+3*(c0-tm*16) : gosub 1550
7516 gosub 6500
7520 sr=4+tm : sc=2+3*(c0-tm*16) : gosub 1550
7530 gosub 6000
7540 return
8000 rem ------------------------
8010 rem Display modified byte at v(c0)
8020 rem ------------------------
8030 v=v(c0) : gosub 1450 : rem convert to HEX
8040 tm=int(c0/16.0001)
8050 sr=4+tm : sc=3+3*(c0-tm*16) : gosub 1550
8060 print v$;
8062 sr=4+tm : sc=31+(c0-tm*16) : gosub 1550
8064 if v(c0)>31 and v(c0)<127 then print chr$(v(c0)); else
print '.';
8070 return
8500 rem ------------------------
8502 rem check for file
8504 rem ------------------------
8510 if fi$<>'' then f=1 : return
8520 e$='No file has been specified. Use "I" command.' : gosub
5000
8530 f=0
8540 return
9000 rem ------------------------
9010 rem error handling routine
9020 rem ------------------------
9030 if err=4 then e$='No such file.' : gosub 5000 : goto 9100
9040 if err=9 then e$='Read error on file.' : gosub 5000 : goto
9100
9050 if err=10 then e$='Write error on file.' : gosub 5000 :
goto 9100
9060 if err=11 then e$='File is write protected.' : gosub 5000 :
goto 9100
9070 if err=12 then e$='Permission flags do not allow access to
file.' : gosub 5000 : goto 9100
9080 if err=34 then e$='Typing CTRL C will not help you.' :
gosub 5000 : goto 9100
9090 on error goto 0

9100 fi$=fi0$ : rem restore file name
9103 if fi$='' goto 9130
9105 if erl=3020 or erl=3750 or erl=3959 then close 1
9107 if erl=3959 then gosub 2000 : gosub 3970
9110 open fi$ as 1 size 1
9120 dim b1,f$(mb-1)=1
9130 resume 3099
```

imtec News

Press Release No. 53

QUIX AND THE IMTEC 256

The IMTEC 256 and QUIX Operating System were designed by the same team to produce a computer which will run software written for UNIX-at high speed and low cost. It supports up to five terminals and a printer with a choice of disk storage options ranging from 20-117 MBYTE (formatted) and a main memory of 256K or 512K.

The IMTEC 256 system comes complete with a five and a quarter inch floppy disk drive and an optional 46 MBYTE tape back-up unit, both built into the chassis.

The whole system is attractively packaged in a matte black box measuring 6 1/2 inches high by 17 inches wide by 12 3/4 inches deep.

An outstanding set of basic software packages are included with every IMTEC 256. These are UNIFLEX Word Processing, SCULPTOR (Formerly SAGE by M.P.D.) Language, and DATABASE Management System, enhanced printer spooling, and asynchronous modem communications.

See our ad in this journal for further details.

***

For further information, contact: Imtec Equipment Inc.
1081 Thomas Busch Memorial Highway
Pennsauken, New Jersey 08110
Phone: Toll-free: 1-800-257-1460
(In New Jersey, phone 609-663-1212)

UNIX is a trademark of Bell Laboratories.

SEPTEMBER, 1984 PAGE 1 of 1

The enclosed modification procedures are for running the General version of TSC FLEX 9 on CREATIVE MICRO SYSTEMS hardware.

Assumptions: ACIA's located at $E3C0, and $E3C8
CMS SYSMON 09 firmware

The following procedures make it possible to run the FLEX 9 Disk Operating System with a minimum of software modifications, while retaining compatability with popular Flex software products, carried by many 68XX vendors.

At the initiation of this project, CMS did not have a 5" controller based on WESTERN DIGITAL firmware, which would be compatable with the Flex DOS. The SOUTHEASTERN DDC-16 was selected since it was compatable. Also it was available in Kit, assembled, or bare board.

The CMS 9618 Proto board was selected for the interface to the Exorciser Bus. The first step was to locate the controller board on the CMS 9618 to insure sufficient clearance for the decoding devices and insertion into the mother board. This position worked out to be..even with the top edge and 3/4" from the right edge. Nylon spacers were used to maintain a 3/8" separation.

The decoding devices and the PIA, used for Generating SWI3 interrupts were located on the remaining clear space of the 9618 proto board. The circuitry used is shown on schematics I and II.

Hardware modifications to the DDC-16: Remove voltage regulators VR 1 & 2. Jumper pins 1 and 3 on same. Place the E2 shorting plug on the 08 position. All other shorting plug positions depend on disk configuration and recommended default positions, explained in the DDC-16 manual. Wiring is point to point and sockets are used for the decoding hardware. The CMS 9612 Proto board has decoding hardware available and could be used to simplify most of the decoding problem. If the 9612 is used, the schematic would have to be altered accordingly. If a controller other than the DDC-16 is used insure that the pin numbering is the same, if not make the necessary adjustments. There is no standardization of S$50 bus functions.

Software modifications: These are compatable with the CMS in house SYSMON firmware, which is preferred to the DEBUG09 (MICROWARE) because it is easier to enter and change data to memory.

Console I/O Driver Package:

```
D3E0.....E700        D377....B7 E3C0
D3E8.....E713        D37B....86 E3C8
D3F3.....F842        D382....86 E3C1
0378.....86 03       D38C....86 E3C9
0372...B7 E3C0       D395....B7 E3C1
0379.....86 16       D39B....86 E3C9
```

There are no modifications required for the Disk Driver package.

General Procedures: Once you have your Console and Disk Driver routines punched into memory, the next step is to enter the Quick Loader routines,(Appendix D, Page 53 of the adaptation Guide). Next jump to $E100, and assuming that you have your Flex Master in Drive 0 you should receive the three Sys-sign prompt, and you are ready to follow the procedures outlined on page 25, section 7, of the adaptation guide.

Avoiding pitfalls: When you write a Command, using the Append Utility, make sure that you add a Transfer Address to the last Binary file in the string. You add this Transfer or Starting address when you create that particular Binary file, so you have to think ahead. If you don't, the DOS will respond with a "CAN'T TRANSFER".

Newdisk: This program allows you initiate an unprogrammed disk so that it can be used by the Flex DOS. The best way to get this program operational is to use the Editor and make the changes listed below. Once the changes are in place then use the Assembler to create your NEWDISK.BIN. Both the Editor and the Assembler are included in the Flex Package. The Manuals are also included.

For 8"disk use as is.
For 5"disk...1BEC...SWKEND IS 1BEA...SWKEND
2FD8...DWKEND IS 1FD4...DWKEND
MAXTRK IS 23 FOR 35 TRACKS
MAXTRK IS 28 FOR 40 TRACKS

COMPLETE ALL OTHER CHANGES REFERENCED ON PAGE 98, APPENDIX G

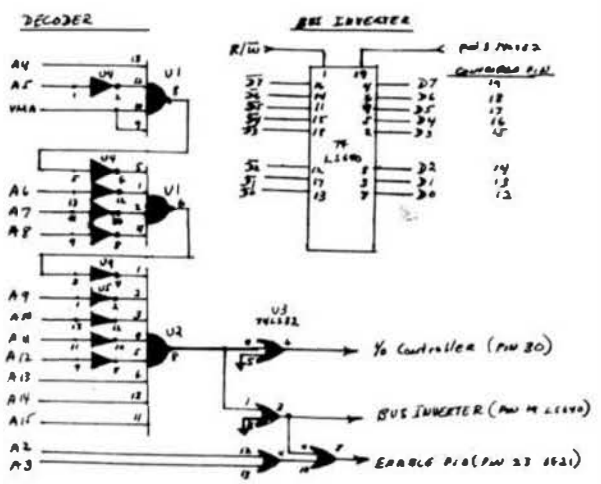
Don't forget to add the Transfer Address ($0100) to the Loader Binary file on Page 87, Appendix O of the adaptation Guide.

Final Steps: You will have to put the program on Page 86 Appendix G into Firmware if you wish to boot from the console. A jump to the selected address is all that is needed, otherwise you will have to input the boot prior to using Flex each time you power on.

Implementing the Printing, and spooling options: It is necessary to adapt PRINT.SYS Drivers for these functions. The software is located on page 3.9 of the Flex Users Manual. Simply change the ACIA addresses. E01C to E3C8 and E01D to E3C9. That is all there is to it. If you wish to tie the Printer to a Parallel Interface, the software for this is located on page 3.7 of the of the same manual. Change the PIA addresses as required. THAT IS ALL THERE IS TO IT!

Anthony J. Gasbarro
23 Center St.
Sullivan, N.H. 03445
Tel: (603) 847-9797

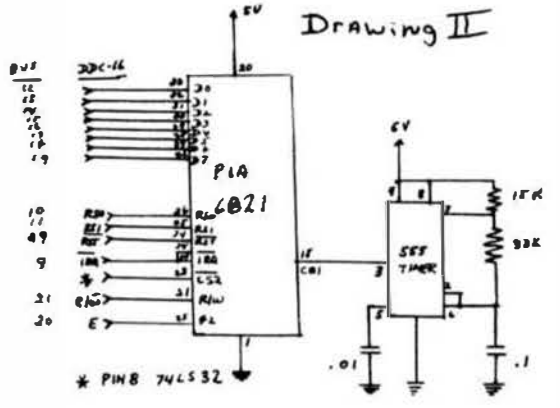
DRAWING I



DECODER, $E01X - DATA BUS INVERTOR

U1 74LS20
U2 74LS30
U3 74LS32

DRAWING II



SWI 3 GENERATOR, 1200 BAUD
FOR SPOOLING IMPLEMENTATION

WINDRUSH
Micro Systems Ltd

Worstead Laboratories (Reg. Office)
North Walsham, Norfolk NR28 9SA
Tel: 06692 404086
Telex: 975548 WMSRO G
MAPCON approved consultants

'68 Micro Journal
Attn: Don Williams
5900 Cassandra Smith
PO Box 849
Hixson, TN 37343
U.S.A.

Your Ref - Our Ref WCD/hb 174 Date 23 Aug 84

Dear Don,

With the recent introduction of our new model Eprom Programmer (UPROM II) into the U.S. market the old model has been discontinued. I would like to clarify the situation in regard to U.S. support of these products by our distributor, GIMIX.

OLD MODEL

FLEX, SSB DOS, MDOS and OS9 Level 1/2 version 1.1 supported. OS9 Level 1/2 version 1.2 is NOT supported. Any queries on this product should be sent to us.

NEW MODEL

1. 6809 FLEX is available directly from us or GIMIX.

2. 6809 SSB DOS is available directly from us ONLY.

3. 6809 MDOS is available directly from us ONLY.

4. OS9 Level 1/2 version 1.2 software is available directly from us or GIMIX.

Since GIMIX only have FLEX and OS9 systems of their own manufacture customers should NOT expect GIMIX to supply or support any other versions of the software, NOR should customers with hardware other than that manufactured by GIMIX expect any support with difficulties arising out of their hardware or software configuration. GIMIX are not in a position to verify any complaints with our product on anything other than GIMIX hardware. We suggest that owners of non-GIMIX hardware contact us if they are not prepared to accept this situation.

We have tested our OS9 software on GIMIX I, II, and III systems and guarantee the software performs as advertised with this hardware. It is simply not practical for us to test OS9 software in all of the hardware environments available. If the software fails to perform the customer must sort it out as it is impossible for us to do so without identical hardware! To this end the assembly language source of the device driver and descriptor are supplied.

Yours faithfully,

William C. Dickinson
DIRECTOR

Here is a review of an exciting new 68000 co-processor. It only works with a Z80 based CP/M 2.2 host, but, even if you bought a new Kaypro II for just this use, it is still the most cost effective 68000 system I'm aware of. CP/M 68k is not the most sophisticated operating system around, but it is quite usable.

I hope this review will encourage some of your readers to consider the 68000 as an alternative to the 6809. It does seem to be the 'coming thing' in microprocessors. Now if MSC could only get OS-9 working on it (this was mentioned in a letter in your June issue, but seems rather far fetched)...

Please note the address change for my subscription.

FROM: DR. MARK BOYD
ST MARY OF THE PLAINS COLLEGE
DODGE CITY KS 67801

TO: DR. MARK BOYD
BOX 83, WSU
WICHITA KS 67208

Thank You,
Mark J. Boyd
Mark J. Boyd

CO - 1668, A 68000 Co- processor Board for CP/M

As a devoted user (since 1975) of Motorola microprocessors, I hated to go over to a Z80 based CP/M system when I needed a portable computer. Reality forced me to get a Kaypro, but my heart stayed with my 6809 OS-9 system. Finally I have found an esthetically satisfying use for my 'utility' CP/M computer, It makes an admirable host for a 68000 co-processor board.

Now don't get me wrong, my Kaypro II is by far the best buy in bundled computer hardware and software around. I have no complaints about the available CP/M 80 software either. Both are far better than their 68xx(x) counterparts. The Z80 may even be a good microcomputer, but it's architecture turns me off. It's just not a pleasure to work with, especially when compared to the 6809.

The 68000 family of microcomputers has the elegant architecture I've come to expect from Motorola. My problem with 68000 systems has been price. Most real 68000 systems seem to be trying to replace small minicomputers, and sell in a similar price range. The single board 68000 systems I've seen at under $1000 are too limited to be useful except for educational purposes. After considerable looking, I finally found a useful 68000 system that I can afford. The heart of this system is a co-processor board known as the CO 1668. It is sold by HSC INC., 262 E. Main Street, Frankfort, New York 13340. Their telephone number is (315) 895-7426. They also have a 8086/8186 co-processor, the CO 1686. Either board works with a wide range of Z80 based, CP/M 2.2 systems.

I have had my CO 1668 for a bit over two weeks now, and have worked with it essentially full time for that period. I'm quite pleased with it. I should also note that HSC was very nice to deal with. I placed my order by phone early in May. They shipped, and billed my VISA card, when they said they would. The salesperson I talked with was reasonably knowledgeable and got answers for my more technical questions quickly. My overall impression of HSC, based on that call and the documentation they wrote for the system, is quite favorable.

A co-processor board has a big price advantage over a full computer system. All it has to provide is a CPU, memory, and one relatively simple I/O channel. The host computer acts as an I/O processor controlling an extensive set of I/O devices. If the host is a mass produced utility computer system, the result is quite cost effective because of the low cost of the host hardware.

In the case of the Kaypro II / CO 1668 combination, a 256K 6MHz 68000 system with parity checking memory, two double density mini floppy drives, a good 80 X 24 display, a good keyboard, a printer port, a RS 232 modem port, and a bunch of software including the "standard" CP/M 8 bit stuff (BASICs, word processing, spread sheets, simple DBMS, and some other utility programs) and CP/M 68K (which includes a decent C compiler, an assembler, a linker, an archive (library manager) program, and the standard CP/M utilities) can be purchased for about $2100 including the Kaypro ($1295). For about $700 more, the 68000 system memory can be expanded to 768K. All the 68000 memory is available as a RAM disk for the CP/M 80 system. Any increment of 128K of this memory can also be used as RAM disk under CP/M 68K. Since both processors use CP/M, the file systems and the console command interpreters are completely compatible. This allows easy switching back and forth between the CPU's, and the use of CP/M 80 utility software with CP/M 68K files. It's not OS-9, but it is a very useable system.

This is sounding a bit too much like an ad. However, the only drawbacks I've discovered in this system are pretty minor. A caveat here, I've only been using the system for a short time, maybe 60 hrs of actual computer time.

Perhaps the most serious drawback: the !@#%^ thing is about 1/2 inch too big too fit conveniently inside the Kaypro. I had to mount it on the outside, at least for now. It requires less than 1 amp at 5 V. which the Kaypro provides without any problem. I am not sure how it would do with the fully expanded memory. Remember this is a general purpose co-processor which will work with any Z80 based system running CP/M 2.2. Also, for non hardware types, HSC sells a separate case and power supply for the co-processor board.

The Kaypro II drives (191 K) are usable, especially in combination with the 128 K RAM disk, but a Kaypro 4 (380 K) would be better. I have had some minor problems with compatibility between CP/M 80 and CP/M 68K, when using CP/M 80 initialized disks with CP/M 68K, the system tracks get written over by data files. This is a minor problem since those tracks are only used when you cold boot the system, warm boots of CP/M 68K don't use the disk. I just keep a separate boot disk and use it only for booting or setting up the RAM disk under CP/M 80. Cold booting is always to CP/M 80, then a program is run to load CP/M 68K and set up the I/O interface. It takes less than half a minute to be up and running CP/M 68K.

To the casual user CP/M 68K is CP/M 80 2.2 with some minor enhancements. For the more serious user, the enhancements are quite significant in that system calls are provided to take advantage of the much more sophisticated 68000 processor. The Digital Research C provided with the operating system does not support floating point, but it is quite powerful and does follow Kernighan and Ritchie. It also provides many of the features of UNIX C, where they make sense in a CP/M 68K environment, or where they can be simulated. I do not have much experience with C, but, after working with it for a week, this does appear to be good implementation. The entire CP/M 68K system seems to be designed around the UNIX C environment, a very good way to go with the 68000.

The software documentation supplied with the system is good but minimal. Several sections assume explicitly that you have other reference books and/or manuals available. This is better than an implicit assumption, but frustrating since these references are neither supplied nor readily available to most users. The hardware documentation is almost nonexistent. The installation is simple and well documented. The overall quality of the documentation is good, but the coverage is barely adequate for an experienced user. There is no tutorial material nor any examples for any of the software other than the installation package. This system is not suitable for an inexperienced user. Familiarity with CP/M and C is assumed. Familiarity with the 68000 is also useful in reading the documentation. The system documentation was designed for hardware neophytes with extensive software backgrounds.

The software supplied with my system came on two IBM format DSDD minifloppies. It was a pain to get it down loaded to Kaypro II format. I think there was a misunderstanding when I ordered, since I requested standard eight inch CP/M. HSC did offer to download to Kaypro II format, but for a fifty dollar charge.

A source code package for HSC's part of the software is mentioned, but price and availability information are not given. HSC says that the supplied source code is set up for their own assemblers. This means a further expense or hassle to make use of it.

I haven't run many benchmarks for performance, but a 6 MHz 68000 with 200 ns. memory should provide more than adequate performance by microcomputer standards (i.e. blow away anything else near this price range), The C version of the sieve algorithm, from BYTE, does ten iterations in about ten seconds. This is several times as fast as the best 8 bit times I've seen for this algorithm. The C compiler uses 3 passes plus the assembly pass and the linker pass. The entire process can be controlled by supplied submit files and takes about two minutes, using the RAM disk for the linker and C library, for small C programs. The output of the linker is a relocatable object file. The archive program is used to build and maintain libraries of functions created as object code files. My overall impression is that CP/M 68K will be a very good program development environment once I get it all figured out. Better documentation would make the figuring out process a bit less time consuming. It's major limitation is the CP/M file structure, but the ability to use my existing file utility software is quite valuable. It is just a single user, single task environment, but it is a decent one.

In conclusion, the CO 1668 co-processor system is a nice piece of work. In conjunction with a Z80 based CP/M 2.2 system, it provides a powerful 68000 system at a very reasonable price. It is expandable, portable in the sense that it could be transferred to other Z80 based systems, and easy to install. It comes with a good software package but weak software documentation. The hardware looks good and works well, but is totally undocumented. Overall, I'm very pleased to have such a nice system for so small a price.

Mark J. Boyd
Box 83, Wichita State University
Wichita, KS 67208

Dear Editor:

Enclosed is a short program for the Bit Bucket. It is a FLEX utility which allows the user to send text directly from the keyboard to the printer. It is called QPRINT. It accepts a line of text from the keyboard using FLEX's INBUFF routine. When "Return" is pressed, QPRINT sends it to the printer. I use this command to do such things as type short notes or address envelopes. INBUFF allows the line of text to be edited using the "Backspace" and "Delete" keys before it is sent to the printer.

QPRINT uses FLEX's POUT vector at $CCE4 (or $ACE4) for the printing. Because of this, it does not turn off Pause like the P command does. The program assumes that the printer drivers have already been loaded into memory. If this in not the case in your system, QPRINT can be APPENDed onto your printer driver so that both will load at the same time.

I hope that the program is of interest to your readers.

Sincerely,

Ken Drexler

Kenneth Drexler

311 Wilson Way
Larkspur, California 94939

```
                        ********************************************
                        *
                        * QUICK PRINT UTILITY
                        *
                        * THIS PROGRAM SENDS TEXT TO THE PRINTER
                        * ONE LINE AT A TIME.  THE INPUT DATA IS
                        * OBTAINED THROUGH INBUFF AND THE "BACKSPACE"
                        * AND "DELETE" KEYS CAN BE USED FOR EDITING.
                        *
                        * DATE:  AUGUST 1, 1984
                        *
                        * FILE NAME: QPRINT.SBC
                        *
                        ********************************************
                        *
                        * SYSTEM EQUATE
                  C000  FLEX    EQU   $C000     FLEX 9
                        * USE $A000 FOR FLEX 2
                        *
                        * EQUATES
                  CCE4  POUT    EQU   FLEX+$0CE4
                  CCC0  PINIT   EQU   FLEX+$0CC0
                  CC09  PSFLAG  EQU   FLEX+$0C09 PAUSE CONTROL
                  CC14  BUFPNT  EQU   FLEX+$0C14
                  CD1B  INBUFF  EQU   FLEX+$0D1B
                  CD1E  PSTRNG  EQU   FLEX+$0D1E
                  CD24  PCRLF   EQU   FLEX+$0D24
                  CD03  WARMS   EQU   FLEX+$0D03
                  CD18  PUTCHR  EQU   FLEX+$0D18
                        ** PROGRAM
C100                          ORG   FLEX+$0100
C100 20  02       QPRINT      BRA   START
                        *
C102 0A           VER         FCB   10          VERSION 1.0
                        * VARIABLE
C103                  PSAVE   RMB   1
                        *
C104 B6  CC09     START       LDA   PSFLAG      TURN OFF PAUSE
C107 B7  C103               STA   PSAVE
C10A 7F  CC09               CLR   PSFLAG
C10D BD  CCC0               JSR   PINIT       INITIALIZE PRINTER
C110 BD  CD24               JSR   PCRLF
C113 8E  C156               LDX   #MSG1       DISPLAY HEADING
C116 BD  CD1E               JSR   PSTRNG
C119 BD  CD24               JSR   PCRLF
C11C BD  CD24     ST???     JSR   PCRLF       END LINE
C11F 86  23                 LDA   #'#         DISPLAY PROMPT
C121 BD  CD18               JSR   PUTCHR
C124 BD  CD18               JSR   PUTCHR
C127 BD  CD18               JSR   PUTCHR
C12A BD  CD1B               JSR   INBUFF
                        *
C12D BD  0B               PRLINE  BSR   PRLINE      PRINT LINE
C12F 26  EB                 BNE   STRT?       NOT EQUAL, CONTINUE
C131 B6  C103               LDA   PSAVE       RESTORE PAUSE
C134 B7  CC09               STA   PSFLAG
C137 7E  CD03               JMP   WARMS
                        *
C13A BE  CC14     PRLINE    LDX   BUFPNT      POINT AT LINE BUFFER
C13D A6  80               PR LB1  LDA   ,X+         GET FIRST CHARACTER
C13F 81  0D                 CMPA  #$0D        CR?
C141 27  12                 BEQ   PRLIN9      YES, EXIT
C143 BD  CCE4               JSR   POUT        NO, PRINT IT
C146 A6  80                 LDA   ,X+         GET NEXT CHARACTER
C148 81  0D                 CMPA  #$0D        CR?
C14A 26  F7                 BNE   PRLIN1      NO, PRINT IT
C14C BD  CCE4               JSR   POUT        YES, PRINT CR, LF
C14F 86  0A                 LDA   #$0A
C151 BD  CCE4               JSR   POUT
C154 4D                     TSTA              SET "NOT EQUAL"
C155 39                     RTS               EXIT
                        *
C156 20 20 51 55  MSG1      FCC   /  QUICK PRINT/
C163 0D 0A 0D 0A            FCB   $D,$A,$D,$A
C167 45 76 65 72            FCC   /Every line is sent to the printer after/
C18E 0D 0A                  FCB   $D,$A
C190 27 52 45 54            FCC   /'RETURN' is pressed.  Exit to FLEX by /
C1B5 0D 0A                  FCB   $D,$A
C1B7 70 72 65 73            FCC   /pressing 'RETURN' at the start of a line./
C1E0 0D 0A                  FCB   $D,$A
C1E2 20 20 20 20 7C         FCC   /   |---------|---------|---------|---------
|---------|---------|/
C221 04                     FCB   4
                        *
                        END   QPRINT

0 ERROR(S) DETECTED
```

GENERAL MICRO SYSTEMS
INCORPORATED

For more information:
George Rigge (714)625-5475

NEW PRODUCT RELEASE

PHOTO ATTACHED

SBC USES 16- OR 8-BIT CPU, HAS 65K MEMORY, MULTIPLE I/O PORTS

ONTARIO, Calif., July 31, 1984 -- An advanced single board computer with a choice of 16- or 8-bit CPUs, 65K of on-board memory, two serial and two parallel ports, an IEEE-488 port, and an Opto 22 port that can be used to directly control industrial relay modules, is now available from General Micro Systems, Inc.

The new module, GMS6307, is the only EXORbus module to offer the 68008 16-bit microprocessor. It also can be supplied with 8-bit 6809, Z80, 6502 or 9900 CPUs in 1 or 2 MHz. Thus, with the 68008, memory operation can be high speed, 8 and 10 MHz, asynchronous. The advanced features can also allow the module to be part of a 2-board 6809 development system. And, systems on the 8-bit data bus can be upgraded to 16-bit when desired.

The module offers extensive digital I/O capability. It includes two full RS232C ports with 15 programmable baud rates. Two parallel printer ports, or one printer and 18 additional user programmable I/O lines, are also offered. A GPIB, or IEEE-488 controller/talker/listener port is also supplied. Additionally, twenty bidirectionally fully buffered I/O lines are each capable of driving 30 mA, for use with industrial I/O modules such as the Opto 22.

65K of on-board, high-speed, static CMOS memory may eliminate any need for additional external memory, reducing system board count. The design allows 68000 memory operation with no wait cycles for faster transfer rates. The memory section provides eight byte-wide sockets which can accept 2K x 8 or 8K x 8 devices, high speed static RAM or ROM/EPROM. Each device may be disabled via DIP switches or the entire memory may be disabled under software control, to allow bootstrapping.

The 6809, Z80, 6502 and 9900 microprocessors are used as part of CPU/translator sets, which can be plugged into the same socket that accepts the 68008. Extended addressing (A16-A19) and bank switching signals (VUA, VMA) are generated when the CPU/translators are used.

Another translator is available to plug into the GPIB socket. This supplies a real time, battery-backed, clock/calendar/RAM in place of the IEEE-488 port. A precision threshold detector and write

protect circuitry is also included.

Priority level interrupt logic eases programming. Power-on reset with an additional reset switch at the top of the module, provides an additional safety feature.

I/O and memory on the CMS6507 are switch selectable for base address, with enable/disable. All address, data and control signals are buffered with tri-state buffers with DMA capability.

The module is fully socketed, over voltage and reverse polarity protected. Burned-in for 72 hours, it carries a full year warranty.

The 6-inch by 9.75-inch CMS6507 advanced single board computer is priced at $685 in single piece quantity (less memory devices). Delivery is from stock, standard OEM discounts are offered.

General Micro Systems Inc., located at 1320 Chaffey Ct., Ontario CA 91762, designs and manufactures a family of microcomputer modules and systems directly compatible with the Motorola Micromodule.EXORciser and Rockwell System 65/AIM 65 busses, plus associated software.

Gerald O'Keefe
2446 Watson Ct.
Palo Alto, CA 94303
(415) 856-0300

CLOCK - MX80 PRINTER INTERFACE

I enclose a schematic and program listing of a clock/MX80 printer interface card I designed. The clock circuit uses the National Semiconductor clock chip MM5817. If have it link in to FLEX™ during bootup, so it displays date and time automatically. The FORMAT for the display is shown below:

DATE THURSDAY SEP 24 1981

TIME 8:49:52

The printer interface to the MX80 printer (which I find is a very good printer for the money) uses 'Centronics' parallel standard, so this should work with other printers also. The software samples the error line from the printer, if an error is found it's printed on the CRT terminal. Printer errors are listed below.

OUT OF PAPER

NOT SELECTED

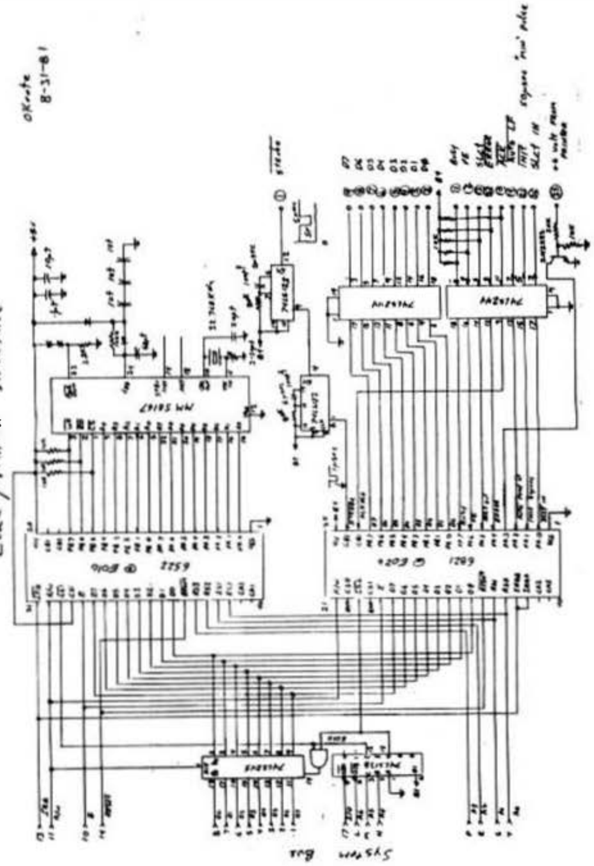
PRINTER POWER OFF

Larry O'Keefe

```
                        *****
                        *****   OPT     PAG
                        *****  EPSON MX-80 PRINTER DRIVER
                        *****   O'KEEFE  AUG 31 81
E020            PIA     EQU     $E020     BASE ADDRESS OF PIA

CD12            OUTCH2  EQU     $CD12     TERMINAL OUTPUT

                        ***** INITALIZE PRINTER INTERFACE PORT
E000                    ORG     $E400
E400 CE 07      PINIT   LDX     #$07      SET DATA DIR OF A PORT
E402 BF E021            STX     PIA+1
E405 8D 0E              BSR     INIT      INITIALIZE PIA PORT A
E407 CE FF              LDX     #$FF      SET FOR 8 BIT OUTPUT
E409 8D 08              BSR     INIT      INITIALIZE PIA PORT B
E40B 86 06              LDA     #$06
E40D B7 E020            STA     PIA
E410 7F E414            CLR     ERFLG     CLEAR PRINTER ERROR FLAG
E413 39                 RTS

E414            ERFLG   RMB     1         PRINTER ERROR FLAG CONTRO

                        *****   INITALIZE PIA PORT
                        *****   X REG IS ADDRESS POINTER
                        *****   B REG IS DATA DIRECTION
E415 86 3E      INIT    LDA     #$3E
E417 A7 84              STA     ,X        PIA+0
E419 30 1F              LEAX    -1,X
E41B E7 80              STB     ,X+       PIA
E41D 86 2C              LDA     #$2C
E41F A7 01              STA     ,X++      PIA+1
E421 39                 RTS

                        ***** SEND CHAR TO PRINTER
                        ***** CHAR IN A REG
                        *****
E422 8D 09      POUT    BSR     PCHK      CHECK PRINTER READY
E424 2A FC              BPL     POUT      SEND TO PRINTER
                        *                         WAIT IF NOT READY
E426 B7 E022            STA     PIA+2
E429 7F E414            CLR     ERFLG     CLEAR PRINTER ERROR FLAG
E42C 39                 RTS

                        ***** CHECK PRINTER READY
                        ***** NEGATIVE FLAG SET IF READY
                        *****
E42D 34 10      PCHK    PSHS    X         SAVE X REG
E42F B6 E020            LDA     PIA       GET PRINTER STATUS
E432 85 08              BITA    #$08      TEST PRINTER PWR BIT
E434 27 02              BEQ     PCHK1     0=ON
E436 8A 08              LDA     #$08      SET POWER FAIL FLAG
E438 85 80      PCHK1   BITA    #$80      PRINTER BUSY?
E43A 26 07              BNE     PCHK2
E43C 7F E414            CLR     ERFLG     CLEAR ERROR FLAG
E43F 1A 08              ORCC    #$08      SET NEGATIVE FLAG
                        *                         PRINTER READY
E441 35 90              PULS    X,PC
E443 85 10      PCHK2   BITA    #$10      TEST FOR ERROR
E445 26 2A              BNE     PCHK      1=ON
E447 7D E414            TST     ERFLG
E44A 26 25              BNE     PCHK
E44C 7C E414            INC     ERFLG
E44F 8E E492            LDX     #MESS1    PRINTER ERROR!
E452 8D 21              BSR     TEXTNL
E454 85 40              BITA    #$40      0=ON
E456 27 05              BEQ     PCHK3
E458 8E E4A5            LDX     #MESS2    OUT OF PAPER
E45B 8D 1A              BSR     TEXT
E45D 85 20      PCHK3   BITA    #$20      TEST FOR SELECTED, 1=NO
E45F 26 05              BNE     PCHK4
E461 8E E4B2            LDX     #MESS3    " NOT SELECTED "
E464 8D 11              BSR     TEXT
E466 81 00      PCHK4   CMPA    #$00      TEST PRINTER CONNECTION
E468 2A 05              BPL     PCHK5
E46A 8E E4C1            LDX     #MESS4    " POWER OFF ON PRINTER"
E46D 8D 08              BSR     TEXT
E46F 8D 13      PCHK5   BSR     CRLF
E471 1E F7      PCHK    EXG     X,X       CLEAR NEGATIVE FLAG
                        *                         PRINTER NOT READY
E473 35 90              PULS    X,PC

                        ***** PRINT TEXT ON TERMINAL
E475 8D 0D      TEXTNL  BSR     CRLF
E477 34 02      TEXT    PSHS    A
E479 A6 80      TEXT1   LDA     ,X+
E47B 27 05              BEQ     OUT
E47D 8D CD12            JSR     OUTCH2
E480 20 F7              BRA     TEXT1
E482 35 82      OUT     PULS    A,PC

                        ***** SEND CR & LF TO TERMINAL
E484 34 02      CRLF    PSHS    A
E486 86 0D              LDA     #$0D
E488 8D CD12            JSR     OUTCH2
E48A 86 0A              LDA     #$0A
E48C 8D CD12            JSR     OUTCH2
E48E 35 82              PULS    A,PC

                        ***** TEXT LIST
E492 20 50 52 49 MESS1  FCC     ' PRINTER ERROR',0
E4A3 20 4F 55 54 MESS2  FCC     ' OUT OF PAPER',0
E4B2 20 4E 4F 54 MESS3  FCC     ' NOT SELECTED',0
E4C1 20 50 52 49 MESS4  FCC     ' PRINTER POWER OFF',0

                        ***** FLEX LINKS
CCCD                    ORG     $CCCD
CCCD 7E E400            JMP     PINIT
CCD0                    ORG     $CCD0
CCD0 7E E420            JMP     PCHK
CCE4                    ORG     $CCE4
CCE4 7E E422            JMP     POUT
                        END

0 ERROR(S) DETECTED
```

```
                        *****
                        *****   OPT     PAG
                        ***** DATE5 PROGRAM
                        *****  O'KEEFE  SEPT 20 81
                        *****
E010            VIA     EQU     $E010     6522 VIA CHIP BASE ADDRESS
E010            IORB    EQU     VIA
E011            IORA    EQU     VIA+1
E012            DDRB    EQU     VIA+2
E013            DDRA    EQU     VIA+3
CD3C            OUTHEX  EQU     $CD3C
CD03            WARMS   EQU     $CD03
CD1B            PUTLHR  EQU     $CD1B
CD24            PCRLF   EQU     $CD24
CD1E            PBYTE   EQU     $CD1E
CCDE            PDATE   EQU     $CCDE     FLEX DATE REGISTER

E500                    ORG     $E500
E500 20 02      START   BRA     TIME      DISPLAY DATE TIME
                        *                         NORMAL ROUTINE START
                        *
E502 20 09              BRA     INIT      START POINT TO SET DATE/TIME
                        *                         OF THE MM5817
                        *****
                        *****
                        ***** TO SET DATE/TIME USE "GET" COMMAND TO BRING
                        ***** DATE PROGRAM INTO MEMORY, THEN USE YOUR
                        ***** MONITOR PROGRAM TO PRESET THE "FCB" DATA
                        ***** BELOW TO THE CORRECT DATE & TIME, NEXT
                        ***** JUMP TO "INIT" ADDRESS THE PROGRAM WILL
                        ***** LOAD THE DATE & TIME THEN DISPLAY IT AND
                        ***** RETURN TO THE FLEX.
E504 01         DATA    FCB     01        SET THOUSANDTHS OF SEC
E505 01                 FCB     01        SET HUNDREDTHS & TENTHS OF S
E506 01                 FCB     01        SET SEC
E507 01                 FCB     01        SET MINUTES
E508 01                 FCB     01        SET HOURS
E509 01                 FCB     01        SET DAY OF WEEK
E50A 01                 FCB     01        SET DAY OF MONTH
E50B 01                 FCB     01        SET MONTH
E50C            TEMP1   RMB     1
E50D            TEMP2   RMB     1
E50E            TADDR   RMB     1
E50F 8D 11      INIT    BSR     INITVIA   INITIALIZE CLOCK'S VIA PORT
E511 17 00CD            LBSR    INTCK     SET CLOCK DATE & TIME
E514 8D 0C      TIME    BSR     INTVIA
E516 17 0023            L       PRDATE
E519 17 004C            LBSR    PRTIME
E51C 17 00B9            LBSR    FLEDAT
                        *                         USED IF CALLED BY FLEX
```

```
                            *                          DATE ROUTINE
E51F 7E      CD03                   JMP    WARMS

                            **** INTIALIZE VIA
E522 7F      E013           INTVIA  CLR    DDRA
E525 86      FF                     LDA    #$FF
E527 39                             RTS


E528 34      02             SPACE   PSHS   A          SEND SPACE TO TERMINAL
E52A 86      20                     LDA    #$20
E52C BD      CD18                   JSR    PUTCHR
E52F 35      02                     PULS   A
E531 39                             RTS

E532 34      02             COLLON  PSHS   A          SEND COLLON
E534 86      3A                     LDA    #$3A       SEND COLON
E536 BD      CD18                   JSR    PUTCHR
E539 35      02                     PULS   A
E53B 39                             RTS

                            ***** PRINT DATE
E53C 8E      E646           PRDATE  LDX    #MESG2
E53F BD      CD1E                   JSR    PSTRNG     PRINT "DATE "
E542 8D      E4                     BSR    SPACE
E544 8E      E67E                   LDX    #DAYS
E547 C6      05                     LDB    #$05
E549 17      00AA                   LBSR   PRTEXT     PRINT DAY OF WEEK
E54C 8D      DA                     BSR    SPACE
E54E 8D      D8                     BSR    SPACE
E550 8E      E64E                   LDX    #MONTHS    PRINT MONTH
E553 C6      07                     LDB    #$07
E555 17      009E                   LBSR   PRTEXT
E558 8D      CE                     BSR    SPACE
E55A C6      06                     LDB    #$06
E55C 8D      40                     BSR    PRREG      PRINT DAY OF MONTH
E55E 8E      E687                   LDX    #YEAR
E561 17      00A3                   LBSR   SCH2       PRINT YEAR
E564 BD      CD24                   JSR    PCRLF
E567 39                             RTS


                            ***** PRINT TIME
E568 8E      E63E           PNTIME  LDX    #MESG1     PRINT TIME
E56B BD      CD1E                   JSR    PSTRNG
E56E 8D      B8                     BSR    SPACE
E570 8E      E500                   LDX    #TEMP2
E573 C6      04                     LDB    #$04
E575 8D      27             LOOP2   BSR    PRREG      PRINT HOUR
E577 8D      B9                     BSR    COLLON
E579 8D      23                     BSR    PRREG      PRINT MINUTES
E57B 8D      B5                     BSR    COLLON
E57D 8D      1F                     BSR    PRREG      PRINT SECONDS
E57F BD      CD24                   JSR    PCRLF
E582 39                             RTS


                            *** LOAD FLEX DATE REG
E583 8E      CC0E           FLEDAT  LDX    #FDATE
E586 C6      07                     LDB    #$07       POINT TO MONTH REGISTER
E588 17      0010                   LBSR   READ       READ MONTH REGISTER
E58B 17      009C                   LBSR   COVBCD

E58E A7      80                     STA    ,X+        SAVE IT
E590 5A                             DECB
E591 17      0014                   LBSR   READ       READ DAY OF MONTH REGISTER
E594 17      0093                   LBSR   COVBCD
E597 A7      80                     STA    ,X+        SAVE IT
E599 86      51                     LDA    #81        LOAD YEAR 81
E59B A7      84                     STA    ,X         SAVE IT
E59D 39                             RTS


                            *** PRINT REG
                            *** B REG HAS CLOCK RE ADDRESS
                            *** DATA RETURNED IN A REGISTER
E59E 8D      08             PRREG   BSR    READ
E5A0 B7      E500                   STA    TEMP2
E5A3 17      006B                   LBSR   PRBCD
E5A6 5A                             DECB
E5A7 39                             RTS

                            **** READ CLOCK REGISTER
                            **** B REG HAS REG ADDRESS
                            **** DATA IN A REGISTER
E5A8 34      04             READ    PSHS   B
E5AA 8D      44                     BSR    ADDR       SEND REG ADDR
E5AC F6      E010                   LDB    IORB       GET ADDR
E5AF C4      BF                     ANDB   #$BF       SEND READ CMD
E5B1 F7      E010                   STB    IORB
E5B4 12                             NOP
E5B5 B6      E011                   LDA    IORA       READ REGISTER VALUE
E5B8 CA      40                     ORB    #$40       RESET RD
E5BA F7      E010                   STB    IORB
E5BD 35      04                     PULS   B
E5BF 39                             RTS

                            ***** WRITE REG
                            ***** A REG=DATA  B REG ADDRESS OF COUNTER
E5C0 34      06             WRITE   PSHS   B,A
E5C2 8D      2C                     BSR    ADDR       SEND CLOCK ADDRESS
E5C4 C6      FF                     LDB    #$FF
E5C6 F7      E013                   STB    DDRA       SET PORT TO OUTPUT
E5C9 B7      E011                   STA    IORA
E5CC F6      E010                   LDB    IORB
E5CF C4      7F                     ANDB   #$7F       SEND WRITE PULSE
E5D1 F7      E010                   STB    IORB
E5D4 12                             NOP
E5D5 CA      80                     ORB    #$80       RESET WRITE PULSE
E5D7 F7      E010                   STB    IORB
E5DA 12                             NOP
E5DB 7F      E013                   CLR    DDRA       PORT A SET TO INPUT
E5DE 35      06                     PULS   B,A
E5E0 39                             RTS

                            **** SET CLOCK REGISTER
E5E1 8E      E506           INTCK   LDX    #DATA+2    POINT TO SET REGISTER

E5E4 C6      02                     LDB    #$02       READ CLOCK REGISTER,SECONDS
E5E6 A6      80             INCK1   LDA    ,X+
E5E8 8D      D6                     BSR    WRITE      WRITE IT TO CLOCK
E5EA 5C                             INCB              INC REGISTER ADDRESS
E5EB C1      08                     CMPB   #$08       LAST REGISTER?
E5ED 26      F7                     BNE    INCK1
E5EF 39                             RTS


                            **** OUTPUT CLOCK REGISTER ADDRESS
E5F0 CA      C0             ADDR    ORB    #$C0       HOLD RD & WR HIGH
E5F2 F7      E010                   STB    IORB
E5F5 39                             RTS

                            *** PRINT TEXT DATE
                            *** B REG HAS CLOCK REG ADDRESS
                            *** X REG HAS TEXT ADDRESS
E5F6 17      FFAF           PRTEXT  LBSR   READ
                            *
E5F9 17      002E           SCHTBL  LBSR   COVBCD     CONVERT BCD TO BINARY
E5FC 1F      89                     TFR    A,B
E5FE 5A                     SCHT1   DECB                SEARCH TABLE FOR TEXT
E5FF 27      06                     BEQ    SCH2
E601 A6      80             SCH1    LDA    ,X+
E603 26      FC                     BNE    SCH1
E605 20      F7                     BRA    SCHT1
E607 A6      80             SCH2    LDA    ,X+        LOAD CHARATOR
E609 27      05                     BEQ    SCH3
E60B BD      CD18                   JSR    PUTCHR     SEND TO TERMINAL
E60E 20      F7                     BRA    SCH2
E610 39                     SCH3    RTS

                            *** PRINT BCD NUMBER
                            *** # IN A REG
E611 34      02             PRBCD   PSHS   A
E613 44                             LSRA
E614 44                             LSRA
E615 44                             LSRA
E616 44                             LSRA
E617 26      05                     BNE    PRBCD1
E619 17      FF0C                   LBSR   SPACE      PRINT SPACE IF REG = 0
E61C 20      02                     BRA    PRBCD2     PRINT REG
E61E 8D      02             PRBCD1  BSR    PRNUB
E620 35      02             PRBCD2  PULS   A
E622 84      0F             PRNUB   ANDA   #$0F
E624 8A      30                     ORA    #$30
E626 BD      CD18                   JSR    PUTCHR     PRINT NUMBER
E629 39                             RTS

                            **** CONVERT BCD BYTE TO BINARY BYTE
                            **** B REG USED AS COUNTER
                            **** DATA IN A REG
E62A 34      04             COVBCD  PSHS   B
E62C 1F      89                     TFR    A,B

E62E C4      0F                     ANDB   #$0F
E630 84      F0                     ANDA   #$F0       SAVE ONLY MSB BYTE
E632 27      06                     BEQ    OUT1
E634 CB      0A             CONV1   ADDB   #10        ADD 10
E636 80      10                     SUBA   #$10
E638 26      FA                     BNE    CONV1
E63A 1F      98             OUT1    TFR    B,A
E63C 35      84                     PULS   B,PC



E63E 20 54 49 4D            MESG1   FCC    ' TIME ',4
E645 04                             FCB    $04

E646 20 44 41 54            MESG2   FCC    ' DATE ',4
E64D 04                             FCB    $04
E64E 4A 41 4E 00            MONTHS  FCC    'JAN',0,'FEB',0,'MAR',0,'APR',0
E65E 4D 41 59 00                    FCC    'MAY',0
E662 4A 55 4E 00                    FCC    'JUN',0,'JUL',0,'AUG',0,'SEP',0,'OCT'
E675 00 4E 4F 56                    FCC    0,'NOV',0,'DEC',0
E67E 53 55 4E 44            DAYS    FCC    'SUNDAY',0,'MONDAY',0,'TUESDAY',0
E694 57 45 44 4E                    'WEDNESDAY',0
E69E 54 48 55 52                    FCC    'THURSDAY',0,'FRIDAY',0,'SATURDAY',0

E6?? 20 31 39 38            YEAR    FCC    ' 1981',0

CA02                                ORG    $CA02
CA02 BD      E500                   JSR    START

                                    END    $CD00

0 ERROR(S) DETECTED
```



406 40th Street
New Orleans, LA 70124
(504) 568-6130

Don Williams
'68' Micro Journal
3018 Hamill Road
P.O. Box 849
Hixson, Tennessee 37343

Dear Don,

I've read with interest the various data encryption articles that have appeared in your magazine. Enciphering a file is useless if the original file is left on a disk. Flex 2.0 users are undoubtably aware that a "deleted" file remains on a disk until its space is reclaimed by the file management system. For those Flex 2.0 users that do not want to encipher files in place, the following program will replace a file with spaces before deleting it:

```
                        * ZERO

                        * AUTHOR:  James L. Dean
                        *          406 40th Street
                        *          New Orleans, La. 70124

                        *     When this source is assembled as
                        * 1.ZERO.CMD, the FLEX 2.0 command
                        * 'i.ZERO d.filename.ext'
                        * will delete the file 'd.filename.ext'
                        * after replacing its contents with
                        * spaces.

                        * EQUATES.

AB40                    FCB     EQU     $AB40       FILE CONTROL BLOCK.
B406                    FMS     EQU     $B406       FILE MANAGEMENT SYSTEM CALL.
AD03                    WARMS   EQU     $AD03       WARMSTART ENTRY POINT.
AD15                    GETCHR  EQU     $AD15       GET CHARACTER.
AD18                    PUTCHR  EQU     $AD18       PUT CHARACTER.
AD2D                    GETFIL  EQU     $AD2D       GET FILE SPECIFICATION.
AD33                    SETEXT  EQU     $AD33       SET EXTENSION.
AD3F                    RPTERR  EQU     $ADJF       REPO T ERROR.

A100                            ORG     $A100

                        * MAIN STARTS HERE.

A100 20 02              LOW     BRA     LOW1
```



```
A102 01                 VN      FCB     1           VERSION NUMBER.

                        * DATA.

A103 04                 DBYTE   FCB     4           NUMBER OF BYTE IN SECTOR
                        *                               CURRENTLY BEING
                        *                               PROCESSED.

                        * ****** PROGRAM EXECUTION STARTS HERE ******

A104 CE AD 40           LOW1    LDX     0CB

                        * OPEN FILE FOR UPDATE.

A107 BD AD 2D                   JSR     GETFIL      GET THE FILE NAME.
A10A 25 0E                      BCS     DSKERR
A10C B6 03                      LDA A   3           SET FOR UPDATE.
A10E A7 00                      STA A   0,X         SAVE IN FILE CONTROL BLOCK.
A110 B6 01                      LDA A   1
A112 BD AD 33                   JSR     SETEXT      SET DEFAULT EXT.
A115 BD B4 06                   JSR     FMS         CALL FILE MANAG MENT SYSTEM.
A118 27 06                      BEQ     LOW2        ERRORS?
A11A BD AD 3F           DSKERR  JSR     RPTERR      REPORT ERROR.
A11D 7E AD 03                   JMP     WARMS       RETURN TO FLEX.
A120 86 FF              LOW2    LDA A   #FF
A122 A7 38                      STA A   59,X

                        * LOOP TO ZERO FILE.

A124 86 00              LOW5    LDA A   0
A126 A7 00                      STA A   0,X
A128 BD B4 06                   JSR     FMS         GET CHARACTER TO BE ZEROED.
A12B 27 03                      BEQ     OKAY7
A12D 7E A1 69                   JMP     LOW7
A130 B6 A1 03           OKAY7   LDA A   DBYTE
A133 A7 23                      STA A   35,X
A135 86 12                      LDA A   18
A137 A7 00                      STA A   0,X
A139 86 20                      LDA A   32
A13B BD B4 06                   JSR     FMS         WRITE SPACE.
A13E 27 06                      BEQ     NEXT2
A140 BD AD 3F                   JSR     RPTERR
A143 7E A1 72                   JMP     NEXT3
A146 B6 A1 03           NEXT2   LDA A   DBYTE
A149 8B 01                      ADD A   1
A14B B1 00                      CMP A   0
A14D 26 14                      BNE     PTROK
A14F CE A8 40                   LDX     0CB
A152 B6 0A                      LDA A   10
A154 A7 00                      STA A   0,X
A156 BD B4 06                   JSR     FMS         BE SURE SECTOR WRITTEN.
A159 27 06                      BEQ     WRTOK
A15B BD AD 3F                   JSR     RPTERR
A15E 7E AD 03                   JMP     WARMS
A161 B6 04              WRTOK   LDA A   4
A163 B7 A1 03           PTROK   STA A   DBYTE
A166 7E A1 24                   JMP     LOW5

                        * END OF LOOP TO ZERO FILE.

                        * CLOSE FILE.

A169 A6 01              LOW7    LDA A   1,X         CHECK ERROR.
A16B B1 0B                      CMP A   B           IS IT EOF?
A16D 27 03                      BEQ     NEXT3
A16F BD AD 3F                   JSR     RPTERR
A172 B6 04              NEXT3   LDA A   4           CLOSE FILE.
A174 A7 00                      STA A   0,X
A176 BD B4 06                   JSR     FMS
A179 27 03                      BEQ     NEXT4
A17B BD AD 3F                   JSR     RPTERR

                        * DELETE FILE.

A17E B6 0C              NEXT4   LDA A   12
A180 A7 00                      STA A   0,X
A182 BD B4 06                   JSR     FMS
A185 27 03                      BEQ     EOJ
A187 BD AD 3F                   JSR     RPTERR
A18A 7E AD 03           EOJ     JMP     WARMS       RETURN TO FLEX.


                                END     LOW
```

I hope that this program solves a security problem for some of your readers.

Sincerely,

James L. Dean

'68' Micro Journal
3018 Hamill Rd.,
PO Box 849
Hixson, Tennessee
37343 U.S.A.

SECTOR.CMD for FLEX 09

Dear Editor:-

Enclosed program is a revised version of Bill Night's SECTOR which appeared in June 1980 issue. Original was modified to fit to FLEX 09 on 5 inch disk and to be position independent.

Reader may have to change following values depending on his system construction.

C10F 01 .. highest drive number in HEX
C11A 28 .. maximum track number in HEX
C125 0A .. highest sector number in HEX

He could disable checking function by replacing these values with FFs.

Many thanks to Bill for his fine work.

Yours truly,

K.Mitadera
126 Sedgefield,
Pointe Claire,
Quebec CANADA
H9R 1N5

```
                        **************
                        * SECTOR.CMD *
                        **************
                        *
                        * THIS PROGRAM ALLOWS ANY SECTOR
                        * ON A DISK TO BE EXAMINED.

                        * THIS IS A REVISED VERSION OF
                        * BILL KNIGHT'S PROGRAM
                        * APPEARED IN JUNE 1980 ISSUE OF 68 MICRO.
                        * FOR FLEX 9.
                        *
                        * TYPE "SECTOR,(DRIVE),(TRACK),(SECTOR)".
                        * ALL NUMBERS IN HEX.

                        * EQUATES.

CD3C                    OUTHEX  EQU     $CD3C
CD42                    GETHEX  EQU     $CD42
```

```
                CD24  PCRLF   EQU    $CD24
                CD18  PUTCHR  EQU    $CD18
                CD03  WARMS   EQU    $CD03
                CD3F  RPTERR  EQU    $CD3F
                CD1E  PSTRNG  EQU    $CD1E

                D406  FMS     EQU    $D406

                C840  FCB     EQU    $C840

                      * ACTUAL PROGRAM STARTS HERE
                      ***************************

C100                          ORG    $C100
C100 20  04           SECT    BRA    SECT1

C102 01               VN      FCB    1          VERSION #1
C103 00               LINE    FCB    0          LINE COUNTER
C104 0000               EMP   FDB    0          TEMPORARY ADDRESS STORAGE

C106 8E  C840         SECT1   LDX    #FCB       POINT TO FCB
C109 17  000C                 LBSR   HEXIN      GET DRIVE #
C10C 2D  73                   BLT    DRVERR     VALID DRIVE?
C10E 81  01                   CMPA   #1         HIGHEST #1 (TWO-DRIVE)
C110 2E  6F                   BGT    DRVERR
C112 A7  03                   STA    3,X        STORE IT IN FCB-DRIVE #
C114 17  0081                 LBSR   HEXIN      GET TRACK #
C117 2D  6E                   BLT    TRKERR     VALID TRACK?
C119 81  27                   CMPA   #39        HIGHEST #39 (5" DISK)
C11B 2E  6A                   BGT    TRKERR
C11D A7  00 1E                STA    30,X       STORE IT IN FCB-TRACK #
C120 BD  76                   BSR    HEXIN      GET SECTOR #
C122 2F  69                   BLE    SECTERR    VALID SECTOR?
C124 81  0A                   CMPA   #10        HIGHEST #10 (5" DISK)
C126 2E  65                   BGT    SECTERR
C128 A7  88 1F                STA    31,X       STORE IT IN FCB-SECTOR
C12B 86  09                   LDA    #9         SET FOR READ SINGLE SECTOR
C12D A7  84                   STA    0,X
C12F BD  D406                 JSR    FMS        OPEN FILE
C132 26  44                   BNE    DSKERR     ERROR?
C134 BD  CD24                 JSR    PCRLF
C137 8E  C85E                 LDX    #FCB+30    OUTPUT TRACK #

C13A BD  6D                   BSR    HEXOUT
C13C 8E  C85F                 LDX    #FCB+31    OUTPUT SECTOR #
C13F BD  68                   BSR    HEXOUT
C141 BD  CD24                 JSR    PCRLF
C144 8E  C880                 LDX    #FCB+6     POINT TO DATA
C147 86  10                   LDA    #16        SET LINE COUNT
C149 A7  8C 07                STA    LINE,PCR
C14C C6  10           SECT2   LDB    #16        SET COLUMN COUNT
C14E AF  8C 03                STX    TEMP,PCR   SAVE X
C151 BD  56           SECT3   BSR    HEXOUT     OUTPUT DATA BYTE
C153 30  01                   LEAX   1,X        POINT TO NEXT DATA
C155 5A                       DECB             DECREMENT COLUMN COUNT
C156 26  F9                   BNE    SECT3      END OF THE LINE?
C158 AE  8C A9                LDX    TEMP,PCR   BEGINNING OF THE LINE
C15B C6  10                   LDB    #16        RESET COLUMN COUNT
C15D A6  88           SECT4   LDA    0,X+       POINT TO DATA
C15F 84  7F                   ANDA   #$7F       MASK MSB
C161 81  1F                   CMPA   #$1F       CONTROL CHR?
C163 22  02                   BHI    SECT5
C165 86  5F                   LDA    #$5F       REPLACE IT BY UNDERSCORE
C167 BD  CD18         SECT5   JSR    PUTCHR     OUTPUT IT
C16A 5A                       DECB             DECREMENT COLUMN COUNT
C16B 26  F0                   BNE    SECT4      END OF THE LINE?
C16D BD  CD24                 JSR    PCRLF
C170 6A  8C 90                DEC    LINE,PCR   DECREMENT LINE COUNT
C1 3 26  D7                   BNE    SECT2      END OF THE SECTOR?
C175 7E  CD03         BKIT    JMP    WARMS

                      * ERROR ROUTINE

C178 30  8D 0035      DSKERR  LEAX   ERR1,PCR   POINT TO MESSAGE
C17C BD  CD1E         REPORT  JSR    PSTRNG
C17F 20  F4                   BRA    EXIT
C181 30  8D 003C      DRVERR  LEAX   ERR2,PCR   POINT TO MESSAGE
C185 20  F5                   BRA    REPORT
C187 30  8D 004B      TRKERR  LEAX   ERR3,PCR   POINT TO MESSAGE
C18B 20  EF                   BRA    REPORT
C18D 30  8D 005A      SECTERR LEAX   ERR4,PCR   POINT TO MESSAGE
C191 20  E9                   BRA    REPORT
C193 8E  CD3F         RDERR   LDX    RPTERR     POINT TO MESSAGE
C196 20  DD                   BRA    EXIT

                      * HEX INPUT

C198 34  10           HEXIN   PSHS   X          SAVE X
C19A BD  CD42                 JSR    GETHEX     GET HEX DATA
C19D 25  F4                   BCS    RDERR      ERROR?
C19F AF  8D FF61              STX    TEMP,PCR   STORE HEX DATA IN TEMP
C1A3 A6  8D FF5E              LDA    TEMP+1,PCR LOAD ITS LSB
C1A7 35  90                   PULS   X,PC       RETURN

                      * HEX OUTPUT

C1A9 BD  CD3C         HEXOUT  JSR    OUTHEX     OUTPUT HEX DATA
C1AC 86  20                   LDA    #$20       OUTPUT A SPACE
C1AE 7E  CD18                 JMP    PUTCHR



                      * ERROR STRINGS

C1B1 44 49 53 48      ERR1    FCC    'DISK READ ERROR'
C1B5 20 52 45 41
C1B9 44 20 45 52
C1BD 52 4F 52
C1C0 04                       FCB    4
C1C1 49 4E 56 41      ERR2    FCC    'INVALID DRIVE NUMBER'
C1C5 4C 49 44 20
C1C9 44 52 49 56
C1CD 45 20 4E 55
C1D1 4D 42 45 52
C1D5 04                       FCB    4

C1D6 49 4E 56 41      ERR3    FCC    'INVALID TRACK NUMBER'
C1DA 4C 49 44 20
C1DE 54 52 41 43
C1E2 4B 20 4E 55
C1E6 4D 42 45 52
C1EA 04                       FCB    4
C1EB 49 4E 56 41      ERR4    FCC    'INVALID SECTOR NUMBER'
C1EF 4C 49 44 20
C1F3 53 45 43 54
C1F7 4F 52 20 4E
C1FB 55 4D 42 45
C1FF 52
C200 04                       FCB    4

                              END    SECT

0 ERROR(S) DETECTED

SYMBOL TABLE:

DRVERR C181   DSKERR C178   ERR1   C1B1   ERR2   C1C1   ERR3   C1D6
ERR4   C1EB   EXIT   C175   FCB    C840   FMS    D406   HEXIN  C198
HEXOUT C1A9   LINE   C103   OUTHEX CD3C   PCRLF  CD24   PSTRNG CD1E
PUTCHR CD18   RDERR  C193   REPORT C17C   RPTERR CD3F   SECT   C100
SECT1  C106   SECT2  C14C   SECT3  C151   SECT4  C15D   SECT5  C167
SECTER C18D   TEMP   C104   TRKERR C187   VN     C102   WARMS  CD03
```

SUPPORT YOUR ADVERTISERS

Classified Advertising

TELETYPE Model 43 PRINTER - with serial (RS232) interface, and full ASCII keyboard. **LIKE NEW** - New cost $1295.00 - ONLY **$759.00** ready to run - Call Tom - Larry - Bob, CPI 615 842-4600

***

MEX6801 Support (development) system for Exorcisor or Exorterm. Consists of Intercept, Control and Buffer modules, software, documentation. User System Evaluator (USE) capability, real time emulation, EXORbus compatible. List price MEX6801 $2700. For sale at $1200. Also 10 card slot power supply, rack mount chassis (M68MMLC) for $300.
Contact Karl Ritzinger (603)-434-2300 (NH) days.

***

Got PC Envy. Selling Gimix 6809 computer with clock, DMA disk controller, 56K RAM. Lots of software including OS-9, FLEX-9. Only $1995. Dual 80tk drives $500.
John Pomeroy (216)372-4457.

***

2 Complete SWPT Sys 6809/6800, MF-68, DC-1, MP-R, MPS, ACT-1, XBASIC, Software, Manuals $700.
Ausle (415) 532-6031.

***

Heathkit Hero-1 Robot with Speech synthesizer. Fully assembled and functions !
Best Offer- (617) 264-4613 evenings.

***

Hard Disk- SWTPC CDS-2 (40MB) assembly, 2 Mhz modification with MPHD2. Just under 2 years old. Make an offer.
Paul Helm, 2520 S. Main St., Akron, OH 44319 (216) 644-2375.

FREE DISKETTE WITH EVERY $50 PURCHASE

Southeast Media

DIET-TRAC Forecaster

DIET-TRAC Forecaster is an XBASIC program that plans a diet in terms of either calories and percentage of carbohydrates, proteins and fats (C P G%) or grams of Carbohydrate, Protein and Fat food exchanges of each of the six basic food groups (vegetable, bread, meat, skim milk, fruit and fat) for a specific individual.

Sex, Age, Height, Present Weight, Frame Size, Activity Level and Basal Metabolic Rate for normal individual are taken into account. Ideal weight and sustaining calories for any weight of the above individual are calculated. When a weight goal is given (either gain or loss), and a calorie plan is agreed upon between the computer and the individual, the number of days to reach the weight goal is projected. The starting and ending rate of weight loss is calculated, and a daily calendar with each day's weight for a 30-day period is printed.

F - $59.95
U - $89.95

Southeast Media

XDATA
A COMMUNICATION Package
for the UniFLEX Operating System

Allows UniFLEX Based Systems to Transmit and Receive files to and from other Computer Systems via Modem. Use with CP/M, Main Frames, other UniFLEX Systems, etc.

-- Verifies Transmission Integrity using checksum or CRC
-- Automatically Re-Transmits bad blocks
-- Transmits data in 128 byte blocks

U - $299.99

Southeast Media

JUST
Text Formatter

JUST, a Text Formatter developed by Ron Anderson, provides numerous features which make it a valuable addition to any FLEX Users Software Library. JUST is designed for formatting Text Output for Dot Matrix Printers and provides many unique features:

-Output the "Formatted" Text to the Display for format analysis and change.
-Output the "Formatted" Text to a Text File for use with the supplied FPRINT.CMD for producing multiple copies of the Text on the Printer INCLUDING IMBEDDED PRINTER COMMANDS (this Utility is very useful at other times also, and worth the price of the program by itself).
-"User Configurable" for adapting to other Printers (comes set up for Epson MX-80 with Graftrax); provides for up to ten (10) imbedded "Printer Control Commands", such as Italics on and off, boldface on and off, etc.
-Automatic compensation for a "Double Width" printed line.
-Includes the normal line width, margin, indent, paragraph, space, vertical skip lines, page length, page numbering, centering, fill, justification, etc.
-Use with ANY Editor.
-Supplied with "Structured Source" (Windrush PL/9); easy to see the flow of the program.

F and CCF - $49.95

Lucidata

PASCAL UTILITIES
Requires LUCIDATA Pascal ver 3.

XREF -- produce a Cross Reference Listing of any text; oriented to Pascal Source.

F and CCF - $25.00

INCLUDE -- allows the inclusion of other Files in a Source Text; has unlimited nesting capabilities. Also allows Binary File inclusions.

F and CCF - $25.00

PROFILER -- produces an Indented, Numbered, "Structogram" of a Pascal Source Text File. Allows viewing the overall structure of large programs, and provides clues as to the integrity of the program. Supplied as Source Code; requires compilation.

F and CCF - $25.00

Lucidata

COPYCAT
Pascal NOT required

Allows reading TSC Mini-FLEX, SSB DOS68, and Digital Research CP/M Disks while operating under FLEX 1.0, FLEX 2.0, or FLEX 9.0 with 6800 or 6809 Systems. COPYCAT will not perform Miracles, but, between the program and the manual, you stand a good chance of accomplishing a transfer. Includes Utilities to List Directories, Copy Files, and convert Text Files when required. Also includes a Utility for investigating Physical Compatibility problems. Programs supplied in Modular Source Code (Assembly Language) to make it easier to solve unusual problems.

F and CCF 5" - $50.00
F 8" - $65.00

Computer Systems Consultants

FLEX DISK UTILITIES

Eight (8) different FLEX Utilities that should be a part of every FLEX Users Toolbox; Assembly Language (Source Code):

Copy a File with CRC Errors, so it can possibly be salvaged; Test Disk for errors; Compare two Disks; a fast Disk Backup Program; Edit Disk Sectors; Linearize Free-Chain on the Disk; print Disk Identification; and Sort and Replace the Disk Directory (in sorted order).

F and CCF - $90.00

WORD PROCESSORS

Alford and Associates

SCREDITOR III

EXTREMELY Powerful Screen-Oriented Editor/Word Processor. Almost 50 different commands; EXCELLENT Documentation (over 300 pages), including a full Tutorial Section to help you learn how to use the system. Features Cursor-based editing, dynamic Screen Formatting (what you see is what you get), Multi-Column display and editing, "decimal align" columns (AND add them up automatically, if wanted), define multiple keystroke macros, even and odd page number headers and footers, imbed printer control codes in text, full justification series of commands, full "help" support, store common command series on disk for future use, etc. Easy "Set-Up" (for example, you just hit the key you want to use for a specific function, such as "cursor up", and the System reads an stores that key - no digging into tech manuals for codes, etc.); use supplied "set-ups", or remap the keyboard to what you are used too. Except for proportional printing, this package will DO IT ALL!

6800 or 6809 FLEX or SSB DOS, OS-9 - $175.00

Great Plains Computer Co.

STYLOGRAPH

A full-screen oriented WORD PROCESSOR -- (now runs on the Data-Comp and FHL Color FLEX Systems; uses the 51 x 24 Display Screens). Full screen display and editing (i.e., what you see is what you get); supports the Daisy Wheel proportional printers.

SPECIAL CCF - $195.00

F and O - $295.00 U - $395.00

SPELL

Fast Computer Dictionary.

F, CCF, OS/9 - $125.00 U - $175.00

MAIL MERGE

Greatly extends the power and flexibility of STYLOGRAPH.

F, CCF, O - $145.00 U - $195.00

VISA MasterCard

"FLEX is a trademark of Technical Systems Consultants
"OS9 is a trademark of Microware

☎ TOLL FREE 1-800-338-6800 For Ordering

SOUTH EAST MEDIA

5900 Cassandra Smith Rd. Hixson, TN 37343

Info (615) 842-4601

CoCo OS-9™ FLEX™ SOFTWARE

Availability Legends —
F = FLEX, CCF = Color Computer FLEX
O = OS-9, CCO = Color Computer OS-9
U = UniFLEX
CCD = Color Computer Disk
CCT = Color Computer Tape

Great Plains Computer Co.

MAIL MERGE

Greatly extends the power and flexibility of SPELLGRAPH. Allows Multiple Text files to be printed out as one large document. Provides for merging information into the Text File during printing (such as different names and addresses), etc.

F, CCF, O - $145.00
U - $195.00

Southeast Media

SPELLB "Computer Dictionary"
OVER 120,000 words!

No more "Let your fingers do the walking through the Dictionary" while you are entering Text with your favorite Editor or Word Processor. SPELLB is more than just "another Spelling Checker"; it allows you to look up a word from within your Editor or Word Processor so that you **KNOW** it is right WHEN YOU TYPE IT IN with the SPH.CMD Utility (which operates in the FLEX Utility Space). Yes, it ALSO allows you to check and update the Text after you are finished; along with allowing you to ADD WORDS to the Dictionary, "Flag" questionable words in the Text for evaluation later, "View a word in context" before changing or ignoring, etc. SPELLB first checks a "Common Word Dictionary", then the normal Dictionary, then a "Personal Word List", and finally, any "Special Word List" you may have specified. SPELLB also allows the use of Small Disk Storage systems.

F and CCF - $129.95

Great Plains Computer Co.

SPELL

Fast Computer Dictionary -- allows directly changing the Text File, adding words to the dictionary, etc. 75,000 words in less than 400 sectors.

F, CCF, OS/9 - $125.00
U - $175.00

DATA BASE MANAGEMENT SYSTEMS

Manchester Applied Business Systems

XDMS

Possibly one of the most powerful Database Management Systems available, this machine language program is small enough to operate on a **single sided 5" disk**, yet provides the **speed** of M.L. and **power** limited only by the user's imagination. This DMS supports Relational, Sequential, Hierarchical, and Random Access File Structures, and has Virtual Memory capabilities for those Giant Data Bases. **XDMS Level I** provides a functional "entry level" System which provides for defining a Data Base, entering and changing the Data, and producing Reports. **XDMS Level II** adds the POWERFUL "**GENERATE**" facility which uses an English Language Command Structure in manipulating the Data to create new File Structures, Sort, Select, Calculate, etc. **XDMS Level III** adds several special "Utilities" which provide additional ease of working with the various structures, changing System Parameters, etc.

XDMS Lvl I - F & CCF - $129.95
XDMS Lvl II - F & CCF - $199.95
XDMS Lvl III - F & CCF - $289.95
XDMS System Manual only - $24.95

Great Plains Computer Co.

DEXPAK DBMS

An XBASIC, Menu Driven, DBMS with "Built-In" Audit Tracking, Extremely Powerful Report & Format Capabilities, etc. This **Time** Proven DBMS will become the "Work Horse" of your Software Stable.

F and CCF $295.00
U $395.00

ACCOUNTING PACKAGES

Great Plains Computer Co. and **Universal Data Research, Inc.** both have Business Packages written in TSC XBASIC for FLEX, **CoCo FLEX**, and **UniFLEX** ----

• • • • • • • • • • • •

FREE DISKETTE WITH EVERY $50 PURCHASE

Computer Systems Consultants

BASIC UTILITY PROGRAMS

Ten BASIC Programs to:

A BASIC **Resequencer** with EXTRAs over "RENUM"; works with ALL Versions of FLEX BASIC **AND** the Precompiler, checks for missing label definitions, processes Disk to Disk instead of in Memory.

Compare, Merge, or Generate Updates between two BASIC Programs, check BASIC **Sequence Numbers**, compare two unsequenced files, and 5 Programs for establishing a Master Directory of several Disks, and sorting, selecting, updating, and printing paginated listings of these files.

A BASIC Cross-**Reference** Program, written in Assembly Language, which provides an X-Ref Listing of the Variables and Reserved Words in TSC BASIC, XBASIC, and PRECOMPILER BASIC Programs.

ALL Utilities include Source (either BASIC or Source Code). An EXCELLENT Value!

F and CCF - $25.00
UniFLEX - $50.00

Computer Systems Consultants

FULL SCREEN INVENTORY/MRP

The Full Screen Inventory System provides a means of maintaining small inventories. Using a linked, keyed random file structure based upon the item field, it keeps the file in alphabetical order for easier inquiry. With the FIND command, the user may locate and/or print all records matching on partial or complete item, description, vendor, or attributes. Items in backorder or below minimum stock levels may be located and/or printed thru the same process. Printed output may be produced in item or vendor order. A materials requirement planning (MRP) capability for manufacturing environments is included to allow the maintenance and analysis of Hierarchical assemblies of items in the inventory file. It requires TSC's Extended BASIC.

F and CCF - $100.00. U - $150.00

The Virginia Company ***Bizpack***

BIZPACK is used for storing accounting, numeric, and financial data which can then be used for planning, budgeting, forecasting, analyzing, etc. While "Electronic Spreadsheets" are extremely useful in many situations, BIZPACK excels in businesses where there are numerous expense columns, revenue sources, significant business indicators, large numbers, erratic week-to-week and month-to-month fluctuations, etc. BIZPACK helps determine statistical relationships, establish trend lines, "smooths" data via moving averages, analyze seasonal data, adjusts for inflation, lags data in Statistics or Column functions, plots data, etc. BIZPACK is oriented toward time series analysis of businesses. The Program displays information on the screen in Columns of Information with each Row conforming to a defined Period of Time (weeks, months, years, etc.), and is very easy to use (data is easy to enter, change, and modify; commands can be renamed to suit the users requirements; unlimited ability to create specialized commands using common BASIC Statements; etc.). Requires TSC's Extended BASIC.

F and CCF - $135.00
with Source - $250.00

*** SPECIAL ***
Purchase XBASIC and BIZPACK together for $221.50
-- a Savings of $13.50 --

VISA MasterCard

"FLEX is a trademark of Technical Systems Consultants
"OS9 is a trademark of Microware

Availability Legends —
F = FLEX, CCF = Color Computer FLEX
O = OS-9, CCO = Color Computer OS-9
U = UniFLEX
CCD = Color Computer Disk
CCT = Color Computer Tape

FREE DISKETTE WITH EVERY $50 PURCHASE

☎ TOLL FREE TELEX 558 414 PVT BTH

1-800-338-6800
For Ordering

SOUTH EAST MEDIA

5900 Cassandra Smith Rd.
Hixson, TN 37343

for information
call (615) 842-4601

CoCo OS-9™ FLEX™

SOFTWARE

*** SPECIAL ***
Purchase KERMIT and EDSPACK together for $221.50
— a Savings of $13.50 —

Computer Systems Consultants
TABULA RASA SPREADSHEET

TABULA RASA is similar to DESKTOP/PLAN and provides for the generation and maintenance of tabular computation schemes often used for analysis of business, sales, and economic scenarios. Its menu-driven user interface provides these capabilities even to those users with no programming experience. Its extensive report-generation capabilities allow the user to generate professional results with minimum effort. It requires TSC's Extended BASIC.

F and CCF - $100.00, U - $125.00

Computer Systems Center
DYNACALC

THE Electronic Spread Sheet for 6809 Computer Systems. An extremely POWERFUL Business Tool, this Program will find an unlimited number of "non-business" applications, also (for example, a Full Junior College Electronics Curriculum was set up using DYNACALC). Advanced features like "Table Lookup" make Income Tax work easy; Column or Row Sorting for numerous applications; etc. Completely "Memory Resident", Machine Language, this Program is FAST. Provides STANDARD FLEX Text File output for use with BASIC, Word Processors, Pascal, "C", etc. Also available for Data-Comp and FHL FLEX systems using the 50 x 24 Displays.

F and SPECIAL OSF - $200.00
CoCo DOS - $99.95
O - $250.00
U - $237.00

ODDS & ENDS

Computer Systems Consultants
FULL SCREEN FORMS DISPLAY

This package supports any Serial Terminal with cursor control or Memory-Mapped Video Displays. The package substantially extends the screen Input/Output capabilities of TSC's Extended BASIC programs by providing a simple, table-driven method of describing and using full screen displays. These table entries are easy to set up and maintain, and are normally stored on disk and read as required. A simple, interactive means of generating the forms and the data field definitions is provided.

F and CCF - $50.00, U - $75.00

Computer Systems Consultants
FULL SCREEN MAILING LIST

The Full Screen Mailing List System provides a means of maintaining simple mailing lists. Using a random fill structure based on the first character of the name field, it maintains the file in alphabetical order for easier inquiry. With the FIND command, the user may locate all records matching on partial or complete name, city, state, zip, or attributes. Printed listings and output to labels may also be produced on the same selective basis. It requires TSC's Extended BASIC.

F and CCF - $100.00, U - $110.00

COLOR COMPUTER SOFTWARE

Stearns Electronics
FORTH

Intrigued by Forth??? Here is a FORTH package tailored to the Color Computer! This package is supplied on Tape, with instructions for transferring it to disk if you wish. Written primarily in machine language, it's speed is unparalleled. A full Semigraphic-6 Editor is provided, along with "goodies" like Graphics and Sound Commands, Printer Commands, Auto-Repeat and Control Keys, etc. If you are interested in learning FORTH, a Trace Feature is provided which is invaluable. If you are a FORTH Pro, this package provides CPU carry Flag accessibility, Fast Task Multiplexing, Clean Interrupt Handling, etc. (Or; you won't "out grow" the Basic capabilities of this Implementation). Combine this package with Leo Brodie's EXCELLENT Book "Starting FORTH", and you will be a FORTH Expert before you know it (and have a lot of fun doing it!).

Color Computer TAPE - $50.95

Custom Software Engineering, Inc.
Color Computer **GRAPHIC SCREEN PRINT** Programs

Dumps any "PMODE" Screen to the Printer with the BASIC USR Function. Shift the Printout Left or Right or Reverse Print (Dark for Light Screen and Vice Versa). All Programs on Tape.

GSPR for R.S. LP-VII/VIII & CMP 100/200/400	$7.95
GSPRE for Epson w/ Graftrax and Graftrax +	$9.95
GSPRG for Gemini 10 and 15	$9.95
GSPRP for the Prowriter Printers	$9.95

Custom Software Engineering, Inc.
DATE-O-BASE CALENDAR Program

A Menu Driven **EXTENDED BASIC** Program which allows the entry of up to 12 Memos per Day, each of which may contain up to 28 Characters, for any day of the Month between the years 1700 and 2099. A Graphic Calendar shows which days contain Memos, and a "Key Word" Search is provided which can be output to the Screen or Printer.

TAPE DATE-O-BASE CALENDAR
(Each Tape File will hold up to 400 Memos) $16.95
DISK DATE-O-BASE CALENDAR
(4,000 Memos at 300/Month per Disk) $19.95

Custom Software Engineering, Inc.
Test's INTEREST-ing

Interested in INTEREST (the Money Kind)? An EXTENDED BASIC Program that will help you deal with numerous problems requiring interest calculations. Present Value, Rate of Return, Current Bond Yield and Rate of Return to maturity, Loan Repayment Amortization Schedules, etc.

TAPE - $29.95

Custom Software Engineering, Inc.
DISK DATA HANDLER 64K

An EXTENDED BASIC Data Management System w/ Mach. Lang. Routines. Allows a max of 246 Chars. and 14 Fields per Record, and another Record can be linked to the first; 8 Char. Field Names, up to 99 Chars. per Field. Powerful On-Screen editor for input and update, flexible Output capabilities including output to Disk Files for use by other Programs. Change File Definition without re-entering the Data, Split Files, etc. Allows Multiple Field Sorts, Select on any combination of Fields, etc. An extremely POWERFUL TOOL; instructions provide examples of Mailing Lists and a Financial Stock Profit and Loss Tracking System.

DISK - $54.95

Custom Software Engineering, Inc.
DISK DOUBLE ENTRY

DISK EXTENDED BASIC Accounting Program w/ Mach. Lang. Routines. A "Traditional" Accounting Package for Small Business, Clubs, Churches, Personal Use, etc. Up to four Levels of subtotals with Trial Balance, Income Statement, and Balance Sheet Reports. DDE allows up to 300 accounts and a Trial Balance of $9,999,999.99. Transactions may be up to 14 lines long, and comments and explanations may be freely used. Accounts are traceable to the journal transaction, which may include comments. Screen reports allow review of past transactions and current balances.

DISK - $44.95

VISA MasterCard

*FLEX is a trademark of Technical Systems Consultants
*OS9 is a trademark of Microware

☎ TOLL FREE
1-800-338-6800
SOUTH EAST MEDIA
5900 Cassandra Smith Rd.
Hixson, TN 37343
info (615) 842-4601
CoCo OS-9™ FLEX™
SOFTWARE

Reliability Legends —
F = FLEX, CCF = Color Computer FLEX
O = OS-9, CCO = Color Computer OS-9
U = UniFLEX
CCD = Color Computer Disk
CCT = Color Computer Tape

PASCAL Compilers

TSC

PASCAL Compiler

Native Code Compiler (UCSD Oriented).

P and CCF - $2[illegible].00

Lucidata

PASCAL Compiler

P-Code Compiler (ISO Standard). Designed especially for Microcomputer Systems; Run-time System checks available resources for each task, allowing operation on even minimal computer systems. Allows linkage to Assembler Code for maximum flexibility.

P and CCF 5" - $190.00
P 8" - $2[illegible].00

OmegaSoft

PASCAL Compiler

For the PROFESSIONAL; ISO Based, Native Code Compiler. Primarily for Real-Time and Process Control applications. Use custom I/O devices in place of the Pascal INPUT and OUTPUT; Long Int. (32 Bit); Dynamic length strings; Interrupt processing, ROM-able, PIC, Re-Entrant Code, etc. POWERFUL! Includes Source for the Symbolic Debugger, Runtime, and several Utilities. Requires a "Motorola Compatible" Relocating Assembler and Linking Loader.

P and CCF - $425.00
One Year Maint. - $1[illegible].00

DECOMPILERS

Southeast Media

DUB (A UniFLEX "basic" De-Compiler)

Re-Create a Source Listing from UniFLEX Compiled basic Programs. Easy to Use; works w/ ALL Versions of UniFLEX basic; Output to Disk or Terminal. Time TESTED and PROVEN; SOLID!

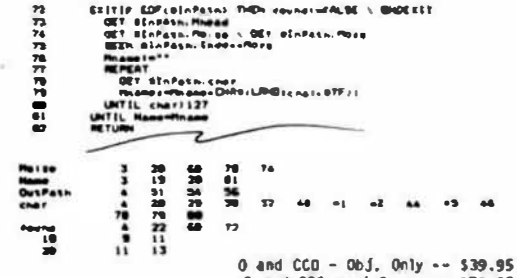
U - $219.95

UTILITIES

Southeast Media

Basic09 XRef

This Basic09 Cross Reference Utility is a Basic09 Program which will produce a "pretty printed" listing with each line numbered, followed by a complete cross referenced listing of all variables, external procedures, and line numbers called. Also included is a Program List Utility which outputs the listing without the overhead of building the cross reference table, which allows it to run considerably faster when only a "pretty printed" listing with line numbers is desired. Requires Basic09 or RunB for operation.



O and CCO - Obj. Only -- $39.95
O and CCO - w/ Source - $79.95

Southeast Media

OS-9 VDisk

Give your OS-9 Level I System the speed of memory access that can be several orders of magnitude over your present floppy disk drive. Use that Extended Memory capability of your SWTPC or Gimix CPU card (or any other that has the same format DAT). The size of the Virtual Disk is completely variable in whole increments of 4K up to 960K, which is all that these systems can address beyond the base page that OS-9 Level I uses. By putting all of your CMDS Directory on your Virtual Disk, you can have the fastest execution speed possible (next to eating up System Memory with all of them). You can also set up high speed inter-process communications via random virtual disk files and not eat up valuable system memory with pipe buffers. Some Assembly Required - Level I ONLY.

O, obj. only - $79.95
w/ Source - $149.95

FREE DISKETTE WITH EVERY $50 PURCHASE

Southeast Media

O-F

---- OS/9 to FLEX - FLEX to OS/9 ----

Finally; the barrier has been removed between OS/9 and FLEX formatted disks! Now you can READ from, and WRITE to, a Single Sided 5" or 8" FLEX diskette from OS-9 with O-F. O-F is a new and unique program, written in BASIC09 (with Source), that performs the following functions;

REFORMAT: A BASIC09 Program that reformats a chosen amount of an OS-9 disk to FLEX Format so it can be used normally by FLEX.

FLEX: A BASIC09 Program that does the actual read or write function to the special O-F Transfer Disk, all selected from a user-friendly menu. Functions provided include reading the FLEX Directory, Deleting FLEX Files, Copying both directions, etc. All selections are interactive and complete, including all necessary prompts to the operator.

FLEX users can read, write and use the special disk as any other FLEX disk, provided the FLEX directory is not allowed to continue beyond track zero (too many files).

F and CCF - $79.95

Southeast Media

COPYMULT

--- Copy LARGE Disks to several smaller disks ---

The following FLEX utilities allow the backup of ANY size disk to any SMALLER size diskettes (Winchester to 8's or 5's, 8" to 5's, etc.). By simply inserting diskettes as requested by COPYMULT, a large disk system may be downloaded to your present floppy disk system, any size. No need to fiddle with directory deletions or any of the other tedious operations that must be done using the normal copy routines.

COPYMULT.CMD understands normal "copy" syntax and always keeps up with files already copied by maintaining directories for both host and receiving disk system, eliminating hours of tedious keyboard entries and other time consuming cleanup chores.

BACKUP.CMD is a special program that downloads "random" type files, any size.

RESTORE.CMD a special program to restructure copied "random" files for copying, or recopying back to the host system.

FREELINK.CMD a "bonus" utility that "relinks" the free chain of floppy or hard disk thereby eliminating fragmentation.

Completely documented source files included.

ALL 4 Programs (8" or 5") $99.50

Southeast Media

CHESS 6809

Requires FLEX and DISPLAYS On Any Type Terminal

Features:

*Four levels of play.
*Swap side. *Point scoring system.
*Two display boards. *Change skill level.
*Solve Checkmate problems in 1-2-3-4 moves.
*Make move and swap sides. *Play white or black.

This is one of the strongest CHESS programs running on any microcomputer, estimated USCF Rating 1600+ (better than most 'club' players at higher levels).

F and CCF - $79.95

VISA MasterCard

*FLEX is a trademark of Technical Systems Consultants
*OS9 is a trademark of Microware

TOLL FREE 1-800-338-6800 For Ordering
SOUTH EAST MEDIA
5900 Cassandra Smith Rd. Hixson, TN 37343
CoCo OS-9™ FLEX™ SOFTWARE
info (615) 842-4601

Availability Legends —

F = FLEX, CCF = Color Computer FLEX
O = OS-9, CCO = Color Computer OS-9
U = UniFLEX
CCD = Color Computer Disk
CCT = Color Computer Tape

ASSEMBLERS

Southeast Media

ASTRUK09

A "Structured Assembler for the 6809" which requires the TSC Macro Assembler. Allows direct use of structured statements such as IF, ELSE, DO, REPEAT, etc., and provides indented level formatting of the listing so that the structure is apparent. Re. '68' Micro Journal, Sept. '83 (program was called "STASM09"; has been renamed due to conflicts).

A User reports

"... I'm very pleased and am now writing almost exclusively in (ASTRUK09). I've selected it over --- for all future systems development... As (one) of my early evaluations, I rewrote a rather elaborate routine originally done in assembly. Out of the 1000 bytes of code generated, the (ASTRUK09) version used only 20 more bytes than the original. --- could not handle this program since it uses triple-precision fixed point arithmetic... I have a large body of code already written that is incompatible with --- constructs. No problem with (ASTRUK09) and the structure sure helps in understanding the logic!"

F, CCF - $99.95

TSC

Macro Assembler

The FLEX STANDARD Assembler. F,CCF $50.00

Relocating Assembler w/Linking Loader

Use with many of the C and Pascal Compilers. F,CCF $150.00

Great Plains Comp. Co.

RMAC

Relocating, Recursive Macro Assembler and Linking Loader.

F,CCF $120.00; w/Source $240.00

OmegaSoft

PRALLI

Relocating Assembler and Linking Loader

F,CCF $125.00; For One Year Maint., add $50.00

Windrush Micro Systems

MACE, by Graham Trott. F,CCF - $98.00

Computer Systems Consultants

SUPER SLEUTH

Computer Systems Consultants Super Sleuth is a "Time Tested", reliable, PROVEN Disassembler that has gained acceptance through out the SS-50 Bus Community as an extremely POWERFUL, INTERACTIVE, Software Tool. The Super Sleuth Software Package consists of 3 Programs; SLEUTH (the Disassembler), CHGNAM (used to globally Change Labels to a meaningful name), and XREF (a Cross Reference Generator for Source Code Files). SLEUTH will Disassemble Memory Resident 6809 Code and 6800, 6801, 6802, 6803 (the "Baby CoCo"), 6805, 6808, 6809, and 6502 (Apple, Atari, Commodore, etc.) Binary Disk Files. (See Aug. '83 '68' Micro Journal "Color Users Notes" Column for a full Review.)

Color Computer	SS-50 Bus (all w/ Source)
CCO (32K Req'd) Obj. Only $49.00	F, $99.00
CCF, Obj. Only $90.00	U, $100.00
CCF, w/Source $99.00	O, $101.00
CCO, Obj. Only $50.00	

ALL Computer Systems Consultants Software
runs on the Color FLEX Systems
ALL in stock
call 800-338-6800
for IMMEDIATE DELIVERY

Computer Systems Center

DynaWare +

An "easy to use", powerful Disassembler for Disk Resident 6809 and 6800 Binary Files. Allows the development of a "Control File" of various Program "Bounderies" during successive disassemblies; can use a Label File which automatically replaces a Hex location with a Label Name; includes an XREF Utility; etc. Label Files provided for Mini-FLEX, FLEX2, FLEX9, Color Computer (for use with Color FLEX Systems), etc. OS-9 Version includes special OS-9 options.

CCF, Obj. Only $1[illegible]0.00
CCO, " " $59.95
F, " " $100.00
O, " " $150.00
U, " " $300.00

COMPILERS & DECOMPILERS

6809 "Structured" Assembly Lang. Compilers

Windrush Micro Systems

PL/9

By Graham Trott. A combination Editor/Compiler/Debugger, all in ONE PACKAGE; provides a totally INTERACTIVE Program Development Cycle. The Single-Pass Compiler supports large Symbol Names; Variable Types; Pointers; Control Structures (similar to 'C' or 'Pascal'); Stack, A- ,B-, and D-Register manipulation; etc. The Source-Oriented Trace/Debugger provides Single Stepping, Breakpointing, etc. An excellent Software Development Tool which provides for the maximum utilization of the power of the 6809.

F, CCF - $1[illegible]

Whimsical Developments

WHIMSICAL

Need the Ease of Design and Maintainability of "Structured Programming" AND the Speed and Control of Assembly Language? Then WHIMSICAL was designed for you! This Single Pass, Recursive Descent Compiler provides the tool for developing simple Utilities to MAJOR Systems in Assembly Language. Supports 3 "Lex" Levels which allow one level of Procedure nesting, or more within "Modules". It is easy to develop programs written for other machines since you are working at the Assembly Language level. Features unified, user-defined I/O; produces ROMable, relocatable, recursive, re-entrant Code; Structured style and statements with Procedures and Modules; supports Byte and Double-Byte primitives with 3 types of Integers (up to 32 bit), Char and Boolean, and unlimited sized Arrays (vectors only); Interrupt handling; unlimited length Variable Names; Variable Initialization (defaults to $00); Include "Source File" directive; Conditional compiling; direct Code insertion; control of the Stack Pointer; etc. To quote Ron Anderson in his review of WHIMSICAL in the Sept. '83 Issue of '68' Micro Journal that, except for the lack of floats, "..., I have to give this one VERY high rating, ...". It is a FAST Compiler which produces FAST Code (his "Primes" Benchmark ran at 9 secs. on a 2 MHz System).

F and CCF - $195.[illegible]

'C' Compilers

Windrush Micro Systems

C Compiler

By James McCosh. Full featured C Compiler for the FLEX Operating System (lacking ONLY "bit-fields"), including an Assembler. Requires the TSC Relocating Assembler IF the user wishes to implement his own Libraries.

F and CCF - $295.00

Introl

C Compiler

A full-featured C, streamlined for the 6809. Generates very efficient object code. Output "benchmarks" close to 10MHz 68000 in 8 Bit Operations; 1.5 times faster than a 4 MHz Z80 when using a 2MHz 6809 System (Re. p 43, '68' Micro Journal, May '83). Floats, etc.

F, CCF, and O - $375.[illegible]
U - $425.00
One Year Maint. - $[illegible]

Availability Legends —
F = FLEX, CCF = Color Computer FLEX
O = OS-9, CCO = Color Computer OS-9
U = UniFLEX
CCD = Color Computer Disk
CCT = Color Computer Tape

Computer Systems Center

OS9SHARE

-- Multi-User, Multi-Tasking with FLEX --

Southeast Media is now shipping OS9SHARE FROM STOCK - the multi-user, multi-tasking capability of OS9SHARE allows FLEX users the advantages of more sophisticated and time saving computer usage without having to buy or learn a new Language or Operating System syntax. OS9SHARE, as its name implies, allows true "time-sharing" operation under the popular FLEX operating system, and also allows each user to run two simultaneous jobs (multi-tasking); even on single-user systems. For example, while in EDIT, you can list another file or examine a directory. Or, you might look up an item in a Data Base while a Sort is in progress! OS9SHARE also provides some fringe benefits that will be greatly appreciated by FLEX users, including type-ahead, command line editing, and instant response to "escape".

OS9SHARE is the painless method! Use your existing Flex computer by simply adding 64K of RAM for each user and/or task. Fact is, you still use FLEX just like you always have! OS9SHARE is not intended as competition to UniFLEX. It does not improve on the speed of FLEX, and does not offer password protection or other niceties of a full-blown multi-user system. **What OS9SHARE does do is give FLEX users a low-cost way to use existing software in a multi-user, multi-tasking environment**, so your existing FLEX versions of BASIC, XBASIC, editors, assemblers, disassemblers, sort/merge packages, word processors, compilers, OS9CALC spread-sheet package, and so on are still good.

NOTE -- The initial release of OS9SHARE is for SWTPC 8/09 Computers, but versions will also be available for other popular extended-memory (up to 1024K) systems, such as HELIX and GIMIX. A minimum of 128K of RAM will be required with ALL versions. OS9SHARE requires **64k of RAM for each active task**; thus a 256k system could allow foreground-background operation on two terminals, or foreground-only operation on four terminals.

AVAILABLE NOW from Southeast Media - $200.00

AUTHORS - PROGRAMMERS
QUALITY SOFTWARE NEEDED
FLEX - UniFLEX - OS/9 - Color Computer

For the past several months, we at t
Southeast Media Division of **Computer Publishi**
Inc. (CPI), the parent company of **'68' MI**
JOURNAL and **COLOR MICRO JOURNAL**, have deba
expanding our software distribution busine
Many other magazines have been doing so
years (in fact, MOST were in the Softwa
Distribution Business BEFORE they began
publish a Magazine). Presently there are m
fine examples of software that has be
developed by YOU, our readers, that will ne
see the "light of day" due to the **Cost**
Advertising and
productio
of that
expos
advert
utili
are
deci
some
ALSO
from
conc
READ
Divis
Distri
Softwar
Programs, et

CRASMB 16.32
6809 Cross Assemble for the 68000 CPU
FLEX & OS-9 $249.00

OSM
6809 Extended Macro Assembler
(Included with K-BASIC)
FLEX & OS-9 $99.00

FREE DISKETTE WITH EVERY $50 PURCHASE

TOLL FREE TELEX 558 414 PVT STH
1-800-338-6800
For Ordering

SOUTH EAST MEDIA

5900 Cassandra Smith Rd.
Hixson, TN 37343
for information
call (615) 842-4601

CoCo OS-9™ FLEX™

SOFTWARE

In the past there has been **too much** software offered that was not quite ready. We will strive to eliminate that element. But, right up front, we tell you only that we will do our very best; nothing more. Also, we will strive to keep cost to a bare minimum, while securing for the author a fair return in
mer

K-BASIC
Basic Language Compiler
(Reviewed in Oct. 1984 68' Micro Journal)
Includes OSM Assembler
Works with CRASMB
FLEX & OS-9 $199.00

CRASMB
6809 Cross Assembler for the following CPU types

6800-2-8	6801-3	6804	6805
6809	6811	6502	1802
8048	8080-5	Z-8	Z-80

FLEX & OS-9 $399.00

will
ualify under this program, please contact one
f the people below. Remember, if your
ftware has any problems or "funnies" -- **GET**
STRAIGHT BEFORE YOU CONTACT US!! Also get
ur source code in proper shape and well
mmented; there is too much 99% code already
ifting around.

If your **software is READY** contact:
Bob Bay, Don Williams, or Tom Williams

Southeast Media is a division
of **Computer Publishing, Inc. (CPI)**,
a family of **100%** 6800X support facilities.

VISA

MasterCard

"FLEX is a trademark of Technical Systems Consultants
"OS9 is a trademark of Microware

TOLL FREE 1-800-338-6800 SOUTH EAST MEDIA 5900 Cassandra Smith Rd. Hixson, TN 37343 info (615) 842-4601 CoCo OS-9™ FLEX™ SOFTWARE

Availability Legend —
F = FLEX, OCF = Color Computer FLEX
O = OS-9, OCO = Color Computer OS-9
U = UniFLEX
OCD = Color Computer Disk
OCT = Color Computer Tape

TEN MOST-ASKED QUESTIONS about DYNACALC™

THE ELECTRONIC SPREAD-SHEET FOR 6809 COMPUTERS

1. What is an electronic spread-sheet, anyway?
Business people use spread-sheets to organize columns and rows of figures. DYNACALC simulates the operation of a spread-sheet without the mess of paper and pencil. Of course, corrections and changes are a snap. Changing any entered value causes the whole spread-sheet to be re-calculated based on the new constants. This means that you can play, 'what if?' to your heart's content.

2. Is DYNACALC just for accountants, then?
Not at all. DYNACALC can be used for just about any type of job. Not only numbers, but alphanumeric messages can be handled. Engineers and other technical users will love DYNACALC's sixteen-digit math and built-in scientific functions. You can build worksheets as large as 256 columns or 256 rows. There's even a built-in sort command, so you can use DYNACALC to manage small data bases — up to 256 records.

3. What will DYNACALC do for ME?
That's a good question. Basically the answer is that DYNACALC will let your computer do just about anything you can imagine. Ask your friends who have VisiCalc™, or a similar program, just how useful an electronic spread-sheet program can be for all types of household, business, engineering, and scientific applications. Typical uses include financial planning and budgeting, sales records, bills of material, depreciation schedules, student grade records, job costing, income tax preparation, checkbook balancing, parts inventories, and payroll. But there is no limit to what YOU can do with DYNACALC.

4. Do I have to learn computer programming?
NO! DYNACALC is designed to be used by non-programmers, but even a Ph.D. in Computer Science can understand it. Even experienced programmers can get jobs done many times faster with DYNACALC, compared to conventional programming. Built-in HELP messages are provided for quick reference to operating instructions.

5. Do I have to modify my system to use DYNACALC?
Nope. DYNACALC uses any standard 6809 configuration, so you don't have to spend money on another CPU board or waste time learning another operating system.

6. Will DYNACALC read my existing data files?
You bet! DYNACALC has a beautifully simple method of reading and writing data files, so you can communicate both ways with other programs on your system, such as the Text Editor, Text Processor, Sort/Merge, STYLOGRAPH™ word processor, RMS™ data base system, or other programs written in BASIC, C, PASCAL, FORTRAN, and so on.

7. How fast is DYNACALC?
Very. Except for a few seldom-used commands, DYNACALC is memory-resident, so there is little disk I/O to slow things down. The whole data array (worksheet) is in memory, so access to any point is instantaneous. DYNACALC is 100% 6809 machine code for blistering speed.

8. Is there a version of DYNACALC for MY system?
Probably. You need a 6809 computer (32k minimum) with FLEX™, UniFLEX™, or OS-9™ operating system. You also need a decent crt terminal, one with at least 80 characters per line, and direct cursor addressing. If your terminal isn't smart enough for DYNACALC, you probably need a new one anyway. The UniFLEX and OS-9 versions of DYNACALC allow you to mix different brands of terminal on the same system. There's also a special version of DYNACALC for Color Computers equipped with FLEX (Frank Hogg or Data-Comp versions).

9. How much does DYNACALC cost?
The FLEX versions are just $200 per copy; UniFLEX version $395; OS-9 version (works with LEVEL ONE or LEVEL TWO) $250. Orders outside North America add $7 per copy for postage. We encourage dealers to handle DYNACALC, since it's a product that sells instantly upon demonstration. Call or write on your company letterhead for more information.

10. Where do I order DYNACALC?
See your local DYNACALC dealer, or order directly from CSC at the address below. We accept telephone orders from 10 am to 6 pm, Monday through Friday. Call us at 314-576-5020. Your VISA or MasterCard is welcome. Please specify diskette size for FLEX or OS-9 versions. Software serial number is required for the UniFLEX version.

Order your DYNACALC today!

Foreign Dealers:

Australia & Southeast Asia: order from Paris Radio Electronics, 161 Bunnerong Road (PO Box 380) Kingsford, 2032 NSW Australia. Telephone: 02-344-9111.

United Kingdom: order from Compusense, Ltd., PO Box 169, London N13 4HT. Telephone: 01-882-0681.

Scandinavia: order from Swedish Electronics hk AB, Murargatan 23-25, Uppsala S-754 37 Sweden. Telephone: 18-25-30-00.

Computer Systems Center
13461 Olive Blvd.
Chesterfield, MO 63017
(314) 576-5020

CSC

UniFLEX software prices include maintenance for the first year.

DYNACALC is a trademark of Computer Systems Center

VisiCalc is a trademark of VisiCorp.
STYLOGRAPH is a trademark of Great Plains Computer Co.
RMS is a trademark of Washington Computer Services.
FLEX and UniFLEX are trademarks of TSC.
OS-9 is a trademark of Microware and Motorola.

WINDRUSH MICRO SYSTEMS

UPROM II

PROGRAMS and VERIFIES: I2758, I2508, I2716, I2516, I2732/2732A, MCM68764/6, I2764/2764A, I2564, I27128/27128A, and I27256. I=Intel, T=Texas, M=Motorola.

NO PERSONALITY MODULES REQUIRED!

TRI-VOLT EPROMS ARE NOT SUPPORTED

INTEL's Intelligent Programming (tm) implemented for Intel 2764, 27128 and 27256 devices. Intelligent programming reduces the average programming time of a 2764 from 7 minutes to 1 minute 15 seconds (under FLEX) with greatly improved reliability.

Fully enclosed pod with 5" of flat ribbon cable for connection to the host computer MC6821 PIA interface board.

MC6809 software for FLEX and OS9 (Level 1 or 2, Version 1.2).

BINARY DISK FILE offset loader supplied with FLEX, MDOS and O S9.

Menu driven software provides the following facilities:

a. FILL ..... a selected area of the buffer with a HEX char.
b. MOVE ..... blocks of data.
c. DUMP ..... the buffer in HEX and ASCII.
d. FIND ..... a string of bytes in the buffer.
e. EXAMINE/CHANGE . the contents of the buffer.
f. CRC ..... checksum a selected area of the buffer.
g. COPY ..... a selected area of an EPROM into the buffer.
h. VERIFY ..... a selected area of an EPROM against the buffer.
i. PROGRAM ..... a selected area of an EPROM with data in the buffer.
j. SELECt ..... a new EPROM type (return to types menu).
k. ENTER ..... the system monitor.
l. RETURN ..... to the operating system.
m. EXECUTE ..... any DOS utility (only in FLEX and OS9 versions).

FLEX AND OS9 VERSIONS AVAILABLE FROM GIMIX. SSB/MDOS CONTACT US DIRECT.

PL/9

- Friendly inter-active environment where you have INSTANT access to the Editor, the Compiler, and the Trace-Debugger, which, amongst other things, can single step the program a SOURCE line at a time. You also have direct access to any FLEX utility and your system monitor.
- 375+ page manual organized as a tutorial with plenty of examples.
- Fast SINGLE PASS compiler produces 8K of COMPACT and FAST 6809 machine code output per minute with NO run-time overheads or license fees.
- Fully compatible with TSC text editor format disk files.
- Signed and unsigned BYTES and INTEGERS, 32-bit floating point REALS.
- Vectors (single dimension arrays) and pointers are supported.
- Mathematical expressions: (+), (-), (*), (/), modulus (%), negation (-)
- Expression evaluators: (=), (<>), (<), (>), (>=), (<=)
- Bit operators: (AND), (OR), (EOR/XOR), (NOT), (SHIFT), (SWAP)
- Logical operators: (.AND), (.OR), (.EOR/XOR)
- Control statements: IF..THEN..ELSE, IF..CASE1..CASE2..ELSE, BEGIN..END, WHILE.., REPEAT..UNTIL, REPEAT..FOREVER, CALL, JUMP, RETURN, BREAK, GOTO.
- Direct access to (ACCA), (ACCB), (ACCD), (BASE), (CCR) and (STACK).
- FULLY supports the MC6809 RESET, NMI, FIRQ, IRQ, SWI, SWI2, and SWI3 vectors. Writing a self-starting (free power-up) program that uses ANY, or ALL, of the MC6809 interrupts is an absolute snap!
- Machine code may be embedded in the program via the 'GEN' statement. This enables you to code critical routines in assembly language and embed them in the PL/9 program (see 'MACE' for details).
- Procedures may be passed and may return variables. This makes them functions which behave as though they were an integral part of PL/9.
- Several fully documented library procedure modules are supplied: IOSUBS, BETID, MATHIO, HEXIO, FLEXIO, SCIPACK, STRSUBS, BASTRING, and REALCON.

'... THIS IS THE MOST EFFICIENT COMPILER I HAVE FOUND TO DATE.'

Quoted from Ron Anderson's FLEX User Notes column in '68. Need we say more?

MACE/XMACE/ASM05

All of these products feature a highly productive environment where the editor and the assembler reside in memory together. Gone are the days of tedius disk load and save operations while you are debugging your code.

- Friendly inter-active environment where you have instant access to the Editor and the Assembler, FLEX utilities and your system monitor.
- MACE can also produce ASMPROCs (GEN statements) for PL/9 with the assembly language source passed to the output as comments.
- XMACE is a cross assembler for the 6800/1/2/3/8 and supports the extended mnemonics of the 6303.
- ASM05 is a cross assembler for the 6805.

D-BUG

LOOKING for a single step tracer and mini in-line disassembler that is easy to use!? Look no further, you have found it. This package is ideal for those small assembly language program debugging sessions. D-BUG occupies less than 6K (including its stack and variables) and may be loaded anywhere in memory. All you do is LOAD If, AIM If and GO! (80 col VDUs only).

McCOSH 'C'

This is as complete a 'C' compiler as you will find on any operating system for the 6809. It is completely compatible with UNIX V7 and only lacks 'bit-fields' (which are of little practical use in an 8-bit world!).

- Produces very efficent assembly language source output with the 'C' source optionally interleaved as comments.
- Built-in optimizer will shorten object code by about 11%.
- Supports interleaved assembly language programs.
- INCLUDES its own assembler. The TSC relocating assembler is only required if you want to generate your own libraries.
- The Pre-Processor, compiler, optimizer, assembler and loader all run independantly or under the 'CC' executive. 'CC' makes compiling a program to executable object as simple as typing in 'CC,HELLO.C <RETURN>'.

IEEE-488

- SUPPORTS ALL PRINCIPAL MODES OF THE IEEE-488 (1975/8) BUS SPECIFICATION:
  - Talker - Serial Poll - Single or Dual Primary Address
  - Listener - Parallel Poll - Secondary Address
  - System Controller - Group Trigger - Talk only ... Listen only
- Fully documented with a complete reprint of the KILOBAUD article on the IEEE bus and the Motorola publication 'Getting aboard the IEEE Bus'.
- Low level assembly language drivers suitable for 6800, 6801, 6802, 6803, 6808 and 6809 are supplied in the form of listings. A complete back to back test program is also supplied in the form of a listing. These drivers have been extensively tested and are GUARANTEED to work.
- Single S-30 board (4, 8 or 16 addresses per port), fully socketed, gold plated bus connectors and IEEE interface cable assembly.

PRICES

D-BUG	(6809 FLEX only)	$ 75.00
MACE	(6809 FLEX only)	$ 75.00
XMACE	(6809 FLEX only)	$ 98.00
ASM05	(6809 FLEX only)	$ 98.00
PL/9	(6809 FLEX only)	$198.00
'C'	(6809 FLEX only)	$295.00
IEEE-488	with IEEE-488 table assembly	$298.00
UPROM-II/U	with one version of software (no cable or interface)	$395.00
UPROM-II/C	as above but complete with cable and S-30 interface	$545.00
CABLE	5' twist-n-flat 50 way cable with IDC connectors	$ 35.00
S-30 INT	S-30 interface for UPROM-II	$130.00
EXOR INT	Motorola EXORbus (EXORciser) interface for UPROM-II	$195.00
UPROM SFT	software drivers for 2nd operating system. Specify FLEX or OS9 AND disk size!	$ 35.00
UPROM SRC	Assembly language source (contact us direct!)	

ALL PRICES INCLUDE AIR MAIL POSTAGE

Terms: CWO. Payment by Int'l Money Order, VISA or MASTER-CARD also accepted.

WORSTEAD LABORATORIES, NORTH WALSHAM, NORFOLK, ENGLAND. NR28 9SA.

TEL: 44 (692) 404086
TLX: 975548 WMICRO G

WE STOCK THE FOLLOWING COMPANIES PRODUCTS: GIMIX, SSB, FHL, MICROWARE, TSC, LUCIDATA, LLOYD I/O, & ALFORD & ASSOCIATES.

FLEX (tm) is a trademark of Technical Systems Consultants, OS-9 (tm) is a trademark of Microware Systems Corporation, MDOS (tm) and EXORciser (tm) are tradeamrks of Motorola Incorporated.

FEATURES THE POWERFUL, THIRD GENERATION, MOTOROLA 6809 PROCESSOR!

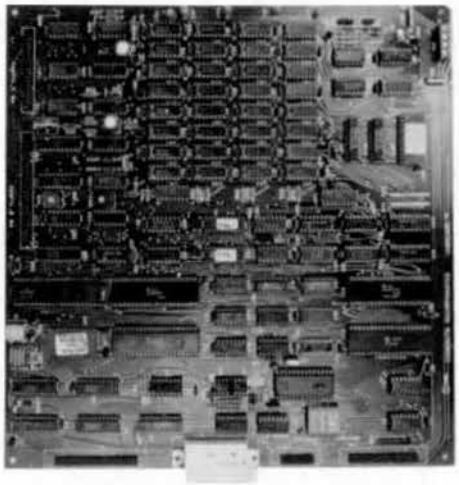
THE 6809 "UNIBOARD"™

SINGLE BOARD COMPUTER KIT

PERFECT FOR COLLEGES, OEM'S, INDUSTRIAL AND SCIENTIFIC USES!

64K RAM! DOUBLE DENSITY FLOPPY DISK CONTROLLER!

New! Lower Price!



BLANK PC BOARD
$99.95
WITH PAL'S, AND TWO EPROMS.

FOR 5-1/4 OR 8 INCH SOURCE DISKETTE ADD $10.

$289.00
COMPLETE KIT! FULLY SOCKETED.

PRICE CUT!!

THE COMPACTA UNIBOARD™: Through special arrangement with COMPACTA INC., we are proud to have been selected the exclusive U.S. Mfg. of their new 6809 UNIBOARD™ COMPUTER KIT. Many software professionals feel that the 6809 features probably the most powerful instruction set available today on ANY 8 bit micro. Now, at last, all of that immense computing power is available at a truly unbelievably low price.

ALL SALES ARE MADE SUBJECT TO THE TERMS OF OUR 90 DAY LIMITED WARRANTY. A FREE COPY IS AVAILABLE UPON REQUEST.

FEATURES:

- ★ 64K RAM using 4116 RAMS.
- ★ 6809E Motorola CPU.
- ★ Double Density Floppy Disk Controller for either 5-1/4 or 8 Inch drives. Uses WD1793.
- ★ On board 80 x 24 video for a low cost console. Uses 2716 Char. Gen. Programmable Formats. Uses 6845 CRT Controller.
- ★ ASCII keyboard parallel input interface. (6522)
- ★ Serial I/O (6551) for RS232C or 20 MA loop.
- ★ Centronics compatible parallel printer interface. (6522)
- ★ Buss expansion interface with DMA channel. (6844)
- ★ Dual timer for real time clock application.
- ★ Powerful on board system monitor (2732). Features commands such as Go To, Alter, Fill, Move, Display, or Test Memory. Also Read and Write Sectors. Boot Normal, Unknown, and General Flex™.

YOUR CHOICE OF POPULAR DISK OPERATING SYSTEMS:

FLEX™ from TSC	$149
OS9™ from Microware	$199

Specify 5-1/4 or 8 Inch

PC BOARD IS DOUBLE SIDED, PLATED THRU SOLDER MASKED, 11 x 11-1/2 IN.

Digital Research Computers
(OF TEXAS)
P.O. BOX 461565 • GARLAND, TEXAS 75046 • (214) 225-2309

TERMS: Shipments will be made approximately 3 to 6 weeks after we receive your order. VISA, MC, cash accepted. Add $4.00 shipping. USA AND CANADA ONLY

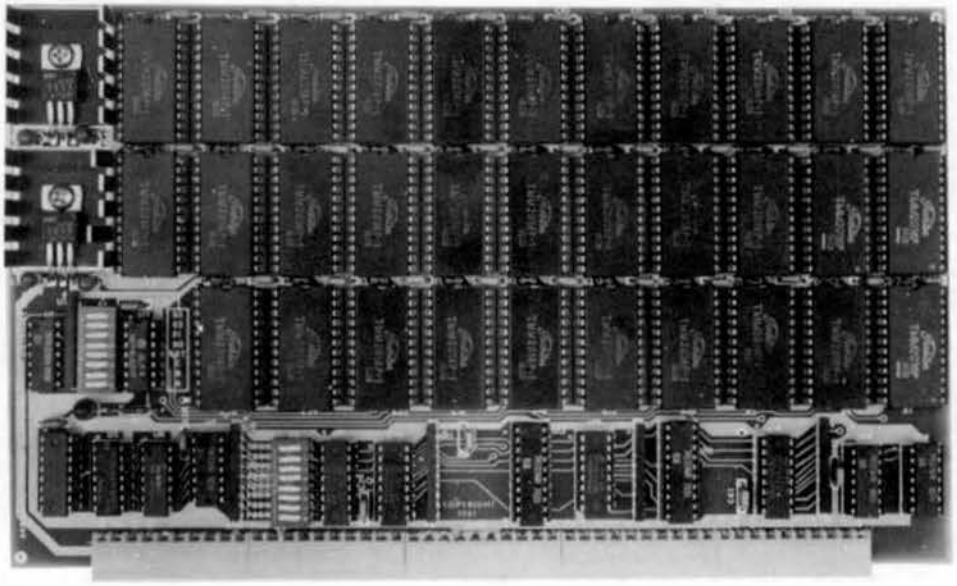
64K SS-50 STATIC RAM

PRICE CUT!!

$159.00 (48K KIT)

NEW!

LOW POWER!



RAM OR EPROM!

BLANK PC BOARD WITH DOCUMENTATION $45

SUPPORT ICs + CAPS -	$18.00
FULL SOCKET SET -	$15.00

ASSEMBLED AND TESTED **ADD $50**

FEATURES:

- ★ Uses new 2K x 8 (TMM 2016 or HM 6116) RAMs.
- ★ Fully supports Extended Addressing.
- ★ 64K draws only approximately 500 MA.
- ★ 200 NS RAMs are standard. (TOSHIBA makes TMM 2016s as fast as 100 NS. FOR YOUR HIGH SPEED APPLICATIONS.)
- ★ Board is configured as 3-16K blocks and 8-2K blocks (within any 64K block) for maximum flexibility.
- ★ 2716 EPROMs may be installed anywhere on Board.
- ★ Top 16K may be disabled in 2K blocks to avoid any I/O conflicts.
- ★ One Board supports both RAM and EPROM.
- ★ RAM supports 2MHZ operation at no extra charge!
- ★ Board may be partially populated in 16K increments.

56K	$189
64K	$219

CLOSE OUT SPECIAL

WE HAVE DROPPED OUR 32K SS-50 STATIC RAM BOARD WHICH USED 2114 LOW POWER RAMS. WE WILL SELL THE REMAINING STOCK OF BLANK PCB'S WITH DATA FOR $17.50 EA. THESE FORMERLY SOLD FOR $50.

16K STATIC RAMS?

The new 2K x 8, 24 PIN, static RAMs are the next generation of high density, high speed, low power, RAMs. Pioneered by such companies as HITACHI and TOSHIBA, and soon to be second sourced by most major U.S. manufacturers, these ultra low power parts, feature 2716 compatible pin out. Thus fully interchangeable ROM/RAM boards are at last a reality, and you get BLINDING speed and LOW power thrown in for virtually nothing.

Digital Research Computers
(OF TEXAS)
P.O. BOX 461565 • GARLAND, TEXAS 75046 • (214) 225-2309

TERMS: Add $2.00 postage. We pay balance. Order under $15 add 75¢ handling. No C.O.D. We accept Visa and MasterCharge. Tex. Res. add 5% Tax. Foreign orders (except Canada) add 20% P & H. Orders over $50, add 85¢ for insurance.

!!! FREE !!!

Published Monthly by Computer Publishing Inc., Hixson, TN.

$1.95

Bulk Rate
U.S. Postage
PAID
Chattanooga, TN
Permit No. 357

Color Micro Journal

The Color Computer Monthly Magazine

$1.95 per issue Vol. 1, Issue 2 October, 1983

THIS 'N THAT

The BIG NEWS this month is that OS-9 has finally arrived for the Color Computer. The ASTOUNDING part of the Radio Shack OS-9 Package, besides the price, is the [illegible]. You 'Old Time Radio Shack Followers' will not believe what you see. Jon Shirley has been telling us that the main reason for the "lack" of documentation with a lot of their products was the restrictions placed on releasing that information by Microsoft: I

OS-9 on the COLOR COMPUTER

One of the "Operating Systems of the Future" is now available for the "little old Color Computer": OS-9. Freely translated, OS-9 means "Operating System for the 6809" (OS-9 is now being written for the [illegible], also). Since it is fairly obvious that UNIX and "UNIX-Type" Operating Systems will be running on just about every computer to come out in the next few years, a whole new language is beginning to appear on the horizon.

Color Computer OS-9, the Package

We had been running a preliminary release of OS-9 on the Color Computer for a few weeks, and received the "Official Radio Shack" version for review a couple of days ago. To put it mildly, this package is IMPRESSIVE! For $69.95 (Radio Shack Catalog Number 26-3038), you receive a 9 1/2" x 7 5/8" x 2" package containing 4

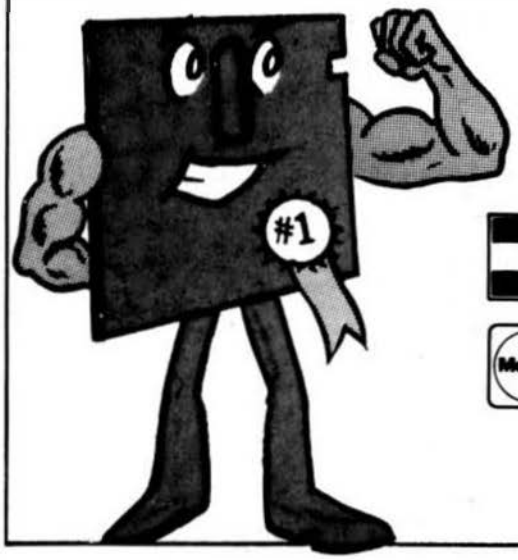
THROW THE BOOKS AWAY . . . ALL of them.

Introducing NuBASE . . . the *uncomplicated* Data Base System from the JBM Group.

NuBASE is a DB manager so versatile that you can use it to do what YOU want with your data. It's not complicated or overbearing; in fact it's so easy to use, you'll be up and running virtually in minutes. Simple user-specified masks insure data accuracy. Data integrity is assured through the use of completely crashproof software. *See what you're doing* through the interactive generation of files, screens and reports.

JBM's NuBASE is as affordable as it is complete. There's nothing "else" to buy . . . $150 brings you the comprehensive package, including a ready-to-use mailing list application to get your NuBASE working for you on *day one.*

The computing power of NuBASE is limited only by the capacity of your hardware.



VISA

MasterCard

currently available for OS-9 Level II

For more information or to place an order, contact:

Dept. 68 14

the JBM group

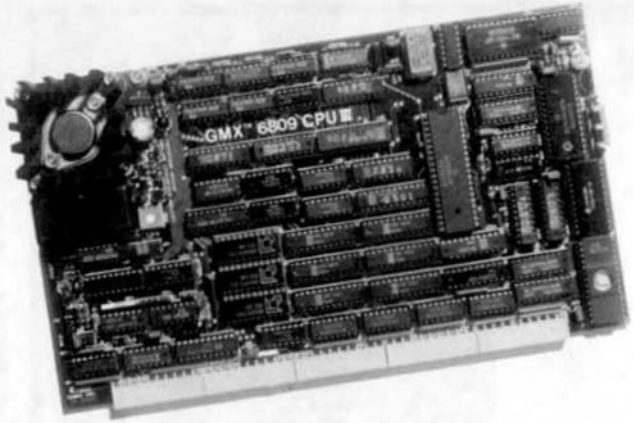
The JBM Group, Inc.
Continental Business Center
Front & Ford Streets
Bridgeport, PA USA 19405
TEL: 215-337-3138
TWX: 510-660-3999

PA res. add 6% sales tax.
US orders, add $5.00 postage and handling.

68' MICRO JOURNAL ADVERTISERS INDEX

This Index is provided as a reader service. The publisher does not assume any liability for omissions or errors.

GIMIX STATE OF THE ART 6809 SYSTEMS FOR THE SERIOUS USER.



GIMIX has 19MB or high performance 47MB Winchester Drive Systems and/or Floppy Disk Drive Systems.

For the user who appreciates the need for a bus structured system using STATIC RAM and powered by a ferro resonant constant voltage transformer.

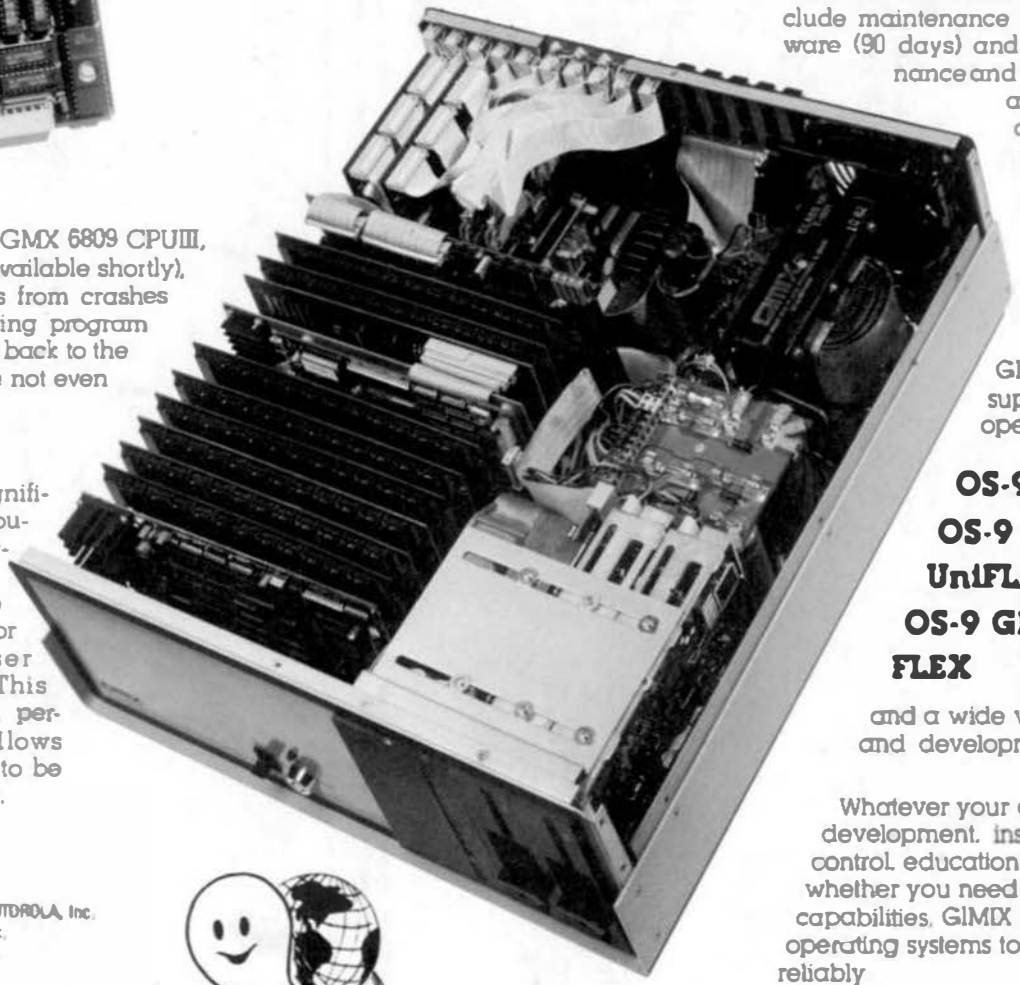
GIMIX has single user systems that can run both FLEX and OS-9 or Multi user systems for use with UniFLEX or OS-9.

GIMIX versions of OS9 and UniFLEX include maintenance and support by Microware (90 days) and TSC (1 year). Maintenance and support after this period are available at extra cost.

(NOTE: this support and maintenance is only for use with approved GIMIX hardware)

'68' Micro Journal

For the ultimate in performance, the Unique GMX 6809 CPUIII, using either OS-9-GMXIII or UniFLEX GMXIII (available shortly), gives protection to the system and other users from crashes caused by defective user programs. e.g. During program development, a programmer who crashes goes back to the shell or the debugger, while the other users are not even aware anything occurred.

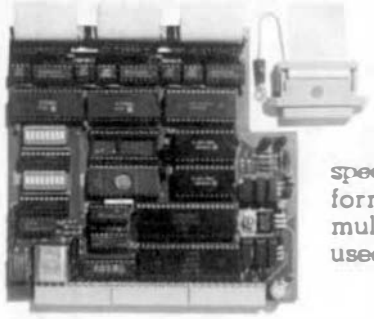


GIMIX 6809 systems support five predominant operating systems:

OS-9 GMX III,
OS-9 GMX II,
UniFLEX,
OS-9 GMX I,
FLEX

and a wide variety of languages and development software.

The intelligent serial I/O processor boards significantly reduce system overhead by handling routine I/O functions, thereby freeing up the host CPU for running user programs. This speeds up system performance and allows multiple terminals to be used at 19.2K baud.



Whatever your application: software development, instrumentation, process control, educational, scientific or business; whether you need single or multi-user capabilities, GIMIX has hardware and the operating systems to get the job done reliably.

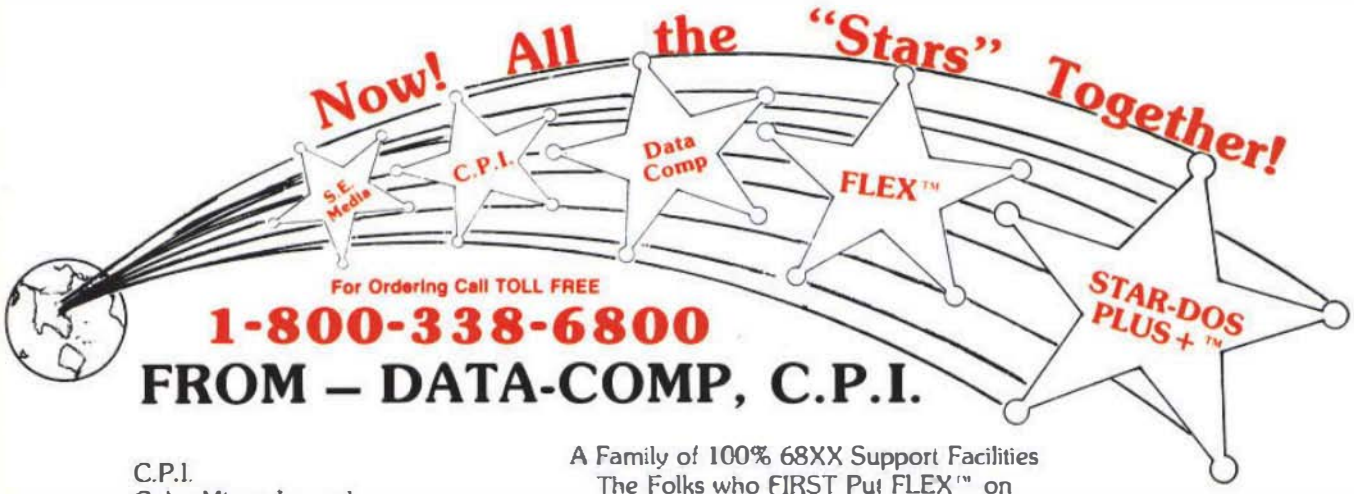
Please phone or write if you need further information.

BASIC-09 and OS-9 are trademarks of Microware Systems Corp. and MOTOROLA, Inc.
FLEX and UniFLEX are trademarks of Technical Systems Consultants, Inc.
GIMIX, GHOST, GMX, CLASSY CHASSIS, are trademarks of GIMIX, Inc.

72

GIMIX INC.

1337 WEST 37th PLACE • CHICAGO, ILLINOIS 60609 • (312) 927-5510 • TWX 910-221-4055



For Ordering Call TOLL FREE

1-800-338-6800

FROM – DATA-COMP, C.P.I.

C.P.I.
Color Micro Journal
'68' Micro Journal
Data-Comp
S.E. Media

A Family of 100% 68XX Support Facilities
The Folks who FIRST Put FLEX™ on
The CoCo
Now Offering: *FLEX™ (2 Versions)
AND *STAR-DOS PLUS+ ™

FLEX-CoCo Sr.
with TSC Editor
TSC Assembler
Complete with Manuals
Reg. $250.00 **Only $79.95**

STAR-DOS PLUS+
- Functions Same as FLEX
- Reads - writes FLEX Disks
- Run FLEX Programs
- Just type: Run "STAR-DOS"
- Over 300 utilities & programs to choose from.

$34.95

FLEX-CoCo Jr.
without TSC Editor & Assembler
$49.95

PLUS

ALL VERSIONS OF FLEX & STAR-DOS• INCLUDE

TSC Editor
Reg $50.00
NOW $35.00

+ Read-Write-Dir RS Disk
+ Run RS Basic from Both
+ More Free Utlities
+ Super 800 Support
+ Free Color Micro Journal 1 yr. sub.
+ External Terminal Program
+ Test Disk Program
+ Disk Examine & Repair Program
+ Memory Examine Program
+ Many Many More!!!

TSC Assembler
Reg $50.00
NOW $35.00

DISK SYSTEMS FOR THE COLOR COMPUTER

THESE PACKAGES INCLUDE DRIVE, *CONTROLLER, POWER SUPPLY & CABINET, CABLE, AND MANUAL.

* SPECIFY WHAT CONTROLLER YOU WANT J&M, OR RADIO SHACK.

PAK #1 - 1 SINGLE SIDED, DOUBLE DENSITY SYS.	$389.95
PAK #2 - 2 SINGLE SIDED, DOUBLE DENSITY SYS.	$639.95
PAK #3 - 1 DOUBLE SIDED, DOUBLE DENSITY SYS.	$439.95
PAK #4 - 2 DOUBLE SIDED, DOUBLE DENSITY SYS.	$699.95
PAK #5 - 2 DOUBLE SIDED, DOUBLE DENSITY SYS. THINLINE DRIVES, HALF SIZE	$659.95
COLOR COMPUTER II 64K W/EXT. BASIC	$189.95

CONTROLLERS

J&M DISK CONTROLLER W/ JDOS OR RADIO SHACK DISK BASIC. SPECIFY WHAT DISK BASIC.	$139.95
RADIO SHACK DISK CONTROLLER 1.1	$134.95

DISK DRIVE CABLES

CABLE FOR ONE DRIVE	$ 19.95
CABLE FOR TWO DRIVES	$ 24.95

MISC

64K UPGRADE W/MOD. INSTRUCTIONS, C,D,E,F, AND COCO 2	$ 49.95
MJL KEYBOARDS	$ 69.95
MICRO TECH LOWER CASE ROM ADAPTER	$ 74.95
RADIO SHACK BASIC 1.2	$ 29.95
RADIO SHACK DISK BASIC 1.1	$ 29.95
RADIO SHACK EXT. BASIC	$ 39.95
SCREEN CLEAN CLEARS UP VIDEO DISTORTION	$ 39.95
MEMOREX DISKS 5" SS,DD	$ 24.00
SHIPPING INCLUDED ON DISK PRICES	
DISK DRIVE CABINET & POWER SUPPLY	$ 49.95
SINGLE SIDED, DOUBLE DENSITY 5" DISK DRIVE	$199.95
DOUBLE SIDED, DOUBLE DENSITY 5" DISK DRIVE	$249.95

PRINTERS

EPSON RX-80	$325.00
EPSON RX-80FT	$375.00
EPSON MX-100	$650.00
EPSON FX-100	$799.00
EPSON FX-80	$549.00
EPSON MX-70	$200.00

SERIAL BOARDS FOR PRINTERS

MX-SERIES	$119.95
FX-SERIES	$ 99.95

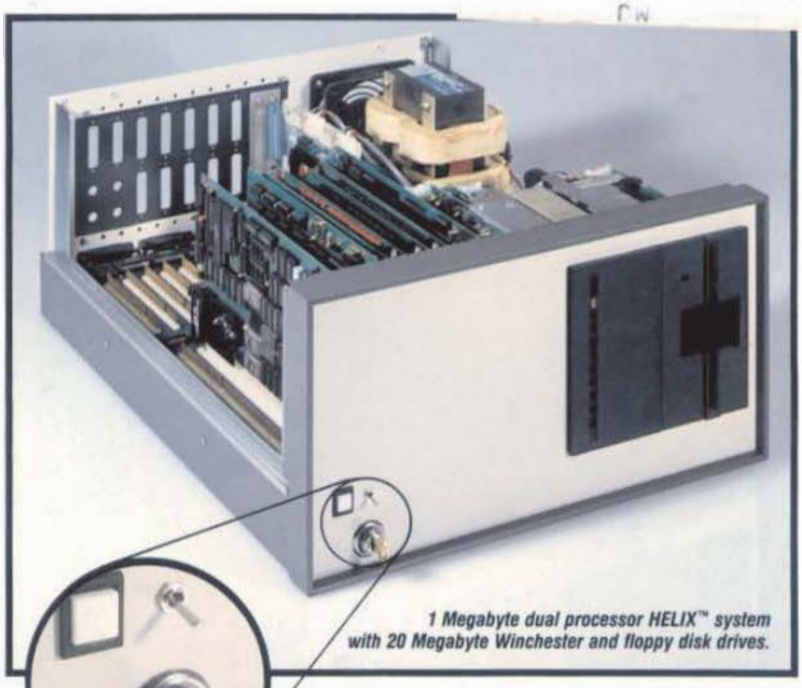
USA ADD 2% SHIPPING
FOREIGN ADD 5% SHIPPING

5900 Cassandra Smith Rd. Hixson, TN 37343

*FLEX is a Trademark of Technical System Consultants
*STAR-DOS + is a Trademark of STAR-Kits & Data-Comp

68 Micro Journal
5900 Cassandra Smith Rd.
Hixson, TN 37343

Second Class Postage Paid
At Chattanooga, TN
ISSN 0194-5025



1 Megabyte dual processor HELIX™ system with 20 Megabyte Winchester and floppy disk drives.

COMPUTER SYSTEMS demonstrates its leadership in computer technology by delivering the only computer system capable of switching between either the 6809 or the 68000 processor. Switching is easily accomplished by a simple front panel toggle switch. The reason we can offer this exclusive feature now, is that when our proven 6809 processor board was designed several years ago, we had the foresight to include the bus controls that allow processor switching.

Hazelwood Computer Systems is also proud to be the first S-50/S-64 bus manufacturer to license and deliver the OS9/68K Operating System from Microware Systems Corporation. OS9/68K is the 68000 version of the popular and powerful OS9 Operating System. Utilizing our proven MC-20 disk controller, OS9/68K can conveniently share a Winchester disk with OS9. Changing from 6809 to 68000 operation is as simple as switching processors and booting the new system from the Winchester disk.

The ease of switching processors and operating systems makes a HELIX™ dual processor system the natural choice for software development. In addition, the advanced design of HELIX™ equipment, emphasizing performance and reliability, makes HELIX™ boards and systems the best value in computing offered anywhere.

System prices vary with configuration. Call for exact pricing.

THE SWITCH IS ON...

The CP-08 processor board utilizes a 68008 processor running at 10 Mhz clock rate. Using proprietary bus synchronization circuitry and single cycle DMA, the CP-08 achieves a marked performance increase over a 2 MHz 6809. Offering absolute compatibility with the 68000 instruction set, the 68008 addresses up to 1 Megabyte of memory. Also included on the CP-08 are up to 4K of ROM, an interrupt timer, and, with battery backup operation, a clock/calendar and 2K RAM. Implemented as a standard S-50 board, the CP-08 brings 68000 operation to S-50 bus computers.

ORDER: CP-08 **PRICE: $595**

The MC-20 Mass Storage Controller board interfaces up to 4 floppy and 8 Winchester disk drives to the S-50/S-64 bus. The MC-20 is an intelligent controller with its own 2 Mhz 6809 processor and 56K RAM. It provides DMA data transfers to a full 24 bit address. All disk operation requests are by logical block number, with the controller performing the necessary track/sector address calculations. Any combination of 5¼ or 8 inch floppy drives can be accommodated with all drive parameters, such as write precompensation, software controlled for each individual drive. Winchester drives are connected via a SASI bus interface. Block address mapping is provided which allows a single drive to be segmented into several logical units. The MC-20 is the controller of the MS-20 Mass Storage Subsystem which includes a 20 Megabyte Winchester drive.

ORDER: MC-20 **PRICE: $695**

OS9/68K

OS9/68K offers increased performance and larger user memory space while retaining all of the features of OS9. Disk file compatibility and operational similarity assures that present OS9 users can easily transfer their operations to the 68000. Included are an editor, assembler, linker, and debugger. A C compiler is available now. BASIC09 and other languages will be avilable soon.

ORDER: OS9/68K **PRICE: $250**

All items available stock to 30 days.
Prices subject to change without notice.

HAZELWOOD COMPUTER SYSTEMS

907 East Terra, O'Fallon, MO 63366, 314-281-1055

HELIX™